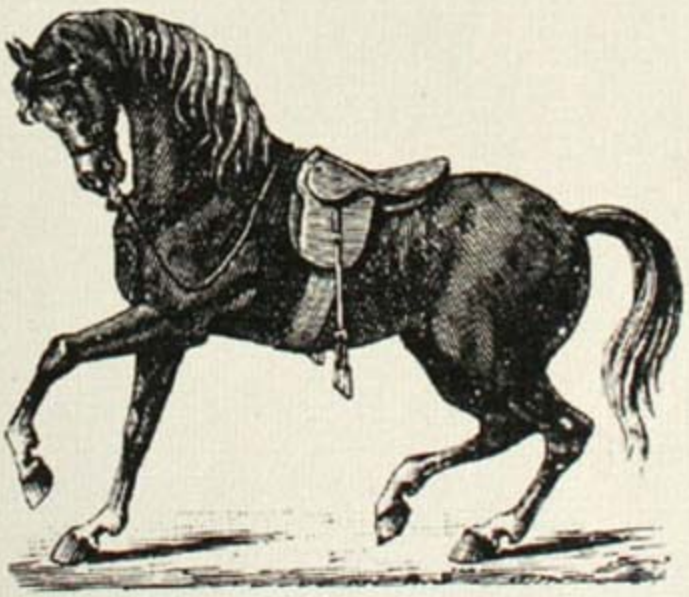
数据库管理系统开发从技术到实践



Delphi 数据库系统开发完全手册

■ 明日科技 宋坤 邹天思 编著



人民邮电出版社
POSTS & TELECOM PRESS

工欲善其事 必先利其器

1

第一篇
Delphi 编程篇

- Delphi 开发环境概述
- 面向对象的应用程序设计
- Delphi 窗体的属性、方法与事件
- 应用程序开发流程
- 用户图形界面设计
- 常用控件的属性、方法与事件
- 菜单栏、工具栏和对话框
- BDE 数据库程序开发技术
- ADO 数据库程序开发技术
- 连接、编辑 Access 数据库
- 连接、编辑 SQLServer 数据库
- 数据控制组件应用
- 调试程序与预错处理

第 1 章　Delphi 开发环境概述
第 2 章　面向对象的应用程序设计
第 3 章　用户图形界面设计
第 4 章　菜单栏、工具栏和对话框
第 5 章　数据库开发技术
第 6 章　访问与编辑数据
第 7 章　数据库用户界面设计
第 8 章　报表与打印程序设计
第 9 章　调试程序与预错处理

第1章 Delphi开发环境概述

Delphi 是由 Borland 公司推出的可视化编程环境，它使用了 Windows 图形用户界面的先进技术，将面向对象的语言功能与方便的可视化开发环境紧密地结合在一起，以其快速的程序开发，强大的数据库处理功能和完善的面向对象表达能力被许多程序员所钟爱。

1.1 Delphi概述

Delphi 最初是作为 Pascal 产品线的修订版出现的，由于在测试期间就表现出许多优秀的特性，Borland 公司决定将其作为单独的工具进行开发，并最终于 1995 年 3 月发布了 Delphi 的第一个版本，即 Delphi 1.0。Delphi 1.0 具有许多非常有用的功能，例如，对数据库和面向对象的强大支持。在 Delphi 的第一个版本推出以后，相继又推出了 Delphi 2、Delphi 3 和 Delphi 4，每一个版本的推出都在上一个版本的基础上增加了许多功能，例如，Delphi 2 提供了对 OLE 自动化服务器的支持，Delphi 3 提供了代码识别技术、组件面板、组件包等，Delphi 4 提供了对 COM 的支持。值得一说的是，Delphi 4 并不是一个很成功的版本，其中的 Bug 比较多，系统不稳定，许多 Delphi 程序员宁可放弃 Delphi 4 提供的新特性而使用备受好评的 Delphi 3。Delphi 5 的出现，弥补了 Delphi 4 中的缺陷，并对集成开发环境（IDE）进行了重大改进。Delphi 6 提供了跨平台（Cross-Platform）组件库（Component Library），使在 Windows 环境下开发的应用程序可以编译、运行在 Linux 操作系统上。Delphi 7 中添加了新技术的支持，例如，IntraWed 的应用、对 Windows.NET 的支持等。

1.2 Delphi 7的集成开发环境

1.2.1 熟悉Delphi 7开发环境

在安装完Delphi 7之后，便可以使用Delphi 7开发应用程序了。首先需要进入Delphi 7编程环境，单击开始菜单中的“所有程序”/“Borland Delphi 7”/“Delphi 7”菜单项，打开Delphi 7开发环境，如图1.1所示。

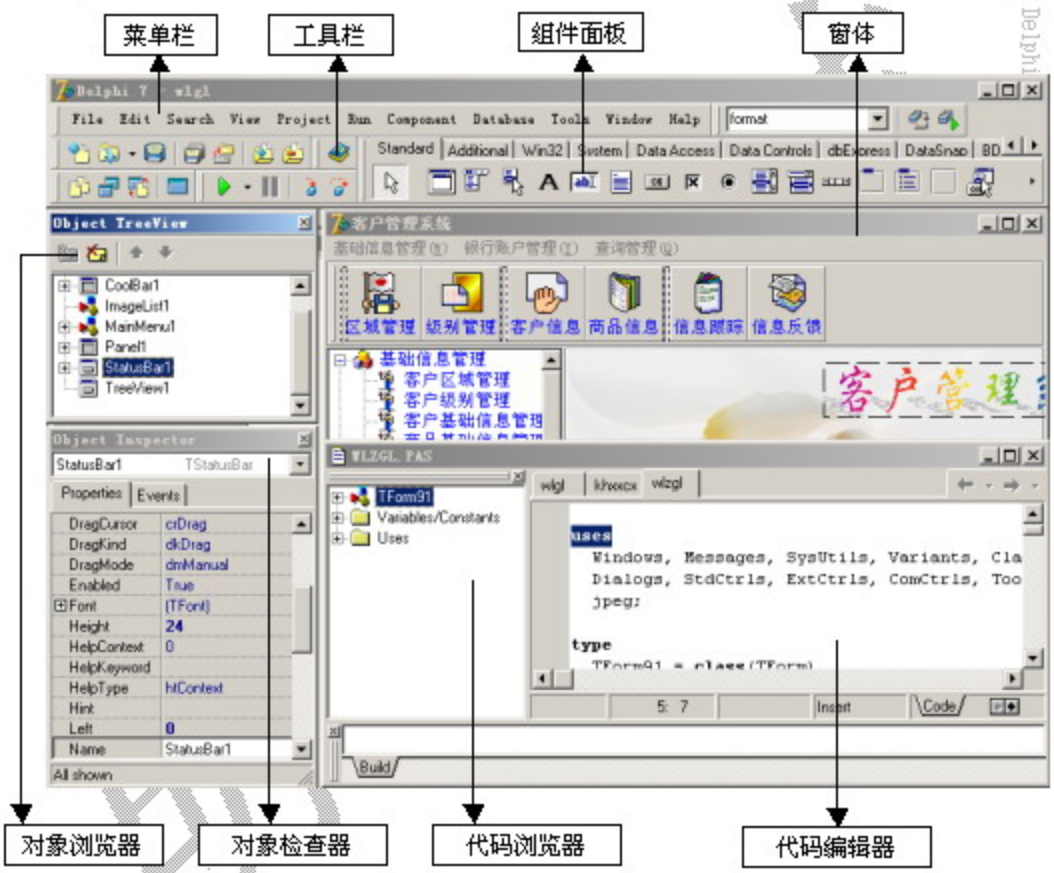


图1.1 Delphi 7开发环境

Delphi 7的编程环境主要由以下几部分组成。

- 菜单栏和工具栏：窗体的最上层是“菜单栏”，菜单栏中提供了Delphi 7集成开发环境的所有功能，而其中一些常用的功能或操作选项则被提取出来放在了“工具栏”中，通过单击这些快捷按钮可以加快程序开发的速度。
- Delphi 7的组件面板：Delphi 7提供了丰富的组件供程序员使用，这些组件都放在组件面板中。整个组件面板分成若干组（页），可选择组标进行选组。Delphi 7提供了约230个组件，分别放在37个页中。
- 窗体：Forms窗口是开展大部分设计的工作区域。可以把部件放在窗体中，通过移动位置、改变尺寸等操作安排控件在窗体中的布局，来开发应用程序

的用户界面。一个应用程序可能不止有一个窗口，可以选用不同的窗体进行设计。其他窗体可以是对话框（Dialog Box）、数据录入框等。当创建一个新项目时，Delphi 7 自动产生一个主窗体。

- 对象浏览器：Object TreeView（对象浏览器），位于 Object Inspector（对象检查器）上方的下拉式菜单中。它显示了窗体上所有部件的名称和类型，也包含窗体本身。用户可以用 Object TreeView 很容易地在窗体的各个部件之间切换，也可以快速地回到窗体本身。当窗体中含有较多的对象时，对象浏览器是切换对象的最快捷途径。
- 对象检查器：对象检查器是联系界面和代码的纽带。主要设置当前对象的初始属性和所涉及的事件处理程序。
- 代码编辑器：代码编辑器是编写程序代码的地方。在 Delphi 中，当部件被放到窗体上时，Delphi 会自动生成大部分的用户界面代码。但要完成特定的功能，还需在代码编辑器中加入完成所需功能的程序段。当程序中含有不止一个窗口时，会有几个库单元的源程序出现在代码编辑器中。代码编辑器的标题条中显示了当前正在编辑的库单元文件名。要查看某一特定程序的源代码，只需用鼠标单击写有该库单元文件名的标签，就可以对该库单元进行编辑了。
- 代码浏览器：代码浏览器以树状图的方式显示代码编辑器中的单元文件信息。它的默认位置在代码编辑器的左边。代码浏览器与代码编辑器配合使用，在代码浏览器中选择某项目，代码编辑器的光标会自动移到所选择的项目代码处。

1.2.2 常用菜单

通过菜单栏可以使用 Delphi 集成开发环境的所有功能，菜单栏中有“文件”、“编辑”、“搜索”、“视图”等 11 个菜单项。单击其中某一个菜单项，便弹出一个相应的菜单。图 1.2 列出了 Delphi 菜单的中英文对照说明。下面介绍开发中最常用的菜单操作。

1. 新建应用程序或窗体

单击“File”/“New”/“Application”选项，将新建一个应用程序。
单击“File”/“New”/“Form”选项，将新建一个应用窗体。

2. 保存应用程序

单击“File”/“Save”选项，将保存当前对程序所做的修改。如果当前文件没有保存过，会弹出保存对话框，要求输入保存的文件名。

单击“File”/“Save As”选项，将当前单元保存为另一个新文件，无论当前单元文件是否保存过，都会弹出一个保存对话框，要求输入保存的文件名。

单击“File”/“Save Project As”选项，将保存当前工程文件为另一个新文件，无论工程文件是否保存过，都会弹出一个保存对话框，要求输入想要保存的文件名。

单击“File”/“Save All”选项，将保存当前所有打开的文件。

3. 打开已存在的文件

单击“File”/“Open Project”选项，将打开“Open Project”对话框，该对话框在默认情况下只显示工程文件，即扩展名为“.dpr”的文件。通过该对话框可以快速地查找并打开工程文件。

Delphi 菜单——中英文对照



图 1.2　Delphi 菜单中英文对照表

Delphi 开发技术教程

单击“File”/“Reopen”选项，会弹出一个下拉菜单，菜单中列出了最近打开过的文件。通过单击文件名可以直接打开这些文件。

4. 关闭应用程序

单击“File”/“Close”选项，将关闭当前的单元文件及窗体，如果当前文件是工程文件，则关闭整个工程。

单击“File”/“Close All”选项，将关闭整个工程文件。

5. 组件置前/置后

单击“Edit”/“Send to Back”选项，将所选择的组件放在最里层，它会被与它重叠的组件所覆盖。

单击“Edit”/“Send to Front”选项，将所选择的组件显示在最外层，它将覆盖其他与它重叠的组件。

6. 锁定组件

单击“Edit/Lock Controls”选项，将锁定窗体中的所有组件，在设计时无法移动组件或改变组件的大小。若想取消对组件的锁定，可以再次单击该选项，将该选项恢复到原来的状态。

7. 打开项目管理器

单击“View”/“Project Manager”选项，将打开项目管理器。项目管理器主要是用来浏览项目组各工程中的文件，通过它可以直接打开或切换工程，而不需要使用关闭、打开的方式切换工程。

8. 打开对象检查器

单击“View”/“Object Inspector”选项，将打开对象检查器，通过对象检查器可以设置当前对象的初始属性和所涉及的事件处理程序。

9. 添加单元文件

单击“Project”/“Add to Project”选项，将打开“Add to Project”工程对话框，通过该对话框可以将单元文件添加到当前的工程中。

10. 运行程序

单击“Run”/“Run”选项，可以运行当前处理的程序。

11. 添加组件单元

单击“Component”/“Install Component”选项，将添加组件单元。在开发组件的过程中，当完成组件的编码后，需要将组件单元放入包中，然后通过编译、安装包将组件安装到开发环境中。

12. 注册 ActiveX 控件

单击“Component”/“Import ActiveX Control”选项，将注册 ActiveX 控件到开发环境中，ActiveX 控件主要是存放在“.ocx”或“.dll”文件中。

明日科技

1.2.3 工具栏

为了方便使用 Delphi 的常用功能，Delphi 7 将菜单栏中一些经常使用的功能放置到工具栏中，供用户选择使用。Delphi 7 的工具栏包括默认工具栏、组建面板（Component Palette）和桌面设定工具（Desktops），Delphi 7 的默认工具栏如图 1.3 所示。

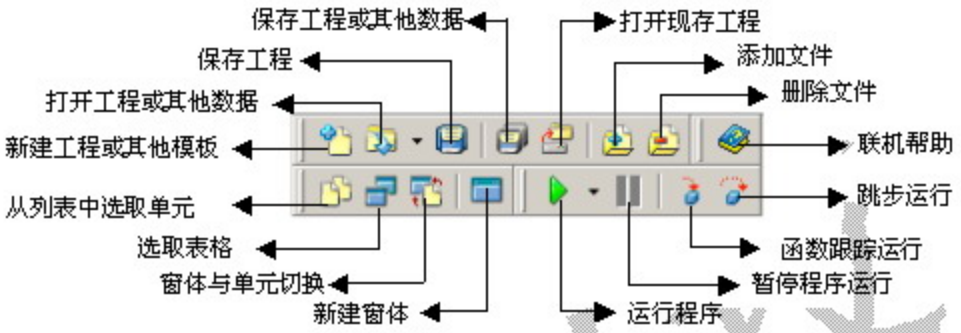


图 1.3 Delphi 7 的工具栏

Delphi 7 的默认工具栏由 4 个部分组成，包括标准工具栏（Standard）、视图工具栏（View）、调试工具栏（Debug）及定制工具栏（Custom），如图 1.4 所示。用户可以根据需要拖曳各个部分自行排列工具栏的位置。



图 1.4 工具栏组成

另外，如果开发人员感觉自己经常使用的功能并不在工具栏上，可以自定义工具栏。用鼠标右键单击工具栏，会弹出一个快捷菜单，如图 1.5 所示。

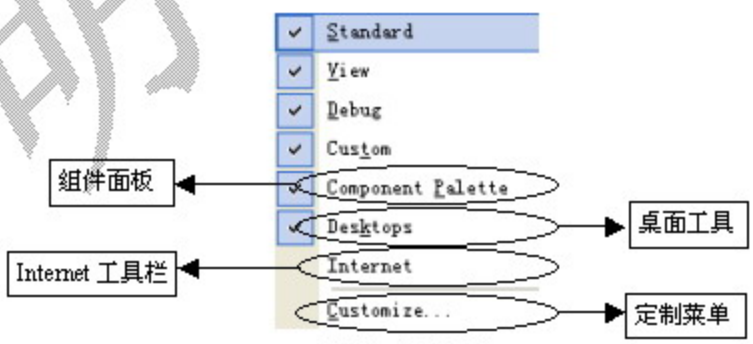


图 1.5 快捷菜单

在快捷菜单中单击“Customize”选项，打开自定义对话框，打开“Commands”选项卡，如图 1.6 所示。

在自定义对话框的“Commands”下拉列表框中选择需要添加的命令（例如 Browser），将它拖曳至 Delphi 工具栏中，如图 1.7 所示。

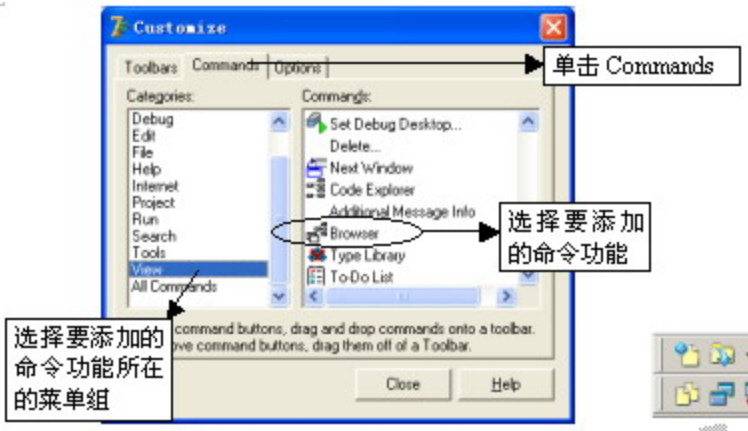


图 1.6　自定义对话框



图 1.7　自定义工具栏

1.2.4　组件面板

组件面板是存放组件的地方，Delphi 定义了组件的标准格式，称之为 VCL（Visual Component Library），组件面板如图 1.8 所示。



图 1.8　组件面板

Delphi 在组件面板上共提供了 19 类组件类型，如表 1.1 所示。

表 1.1　　组件类型

组件名称	说　　明
Standard	标准的组件面板
Additional	附加的组件面板
Win32	Win32 相关的组件面板
System	系统相关的组件面板
Data Access	数据访问组件的组件面板
Data Controls	数据控制组件的组件面板
dbExpress	一组可跨 Linux 平台的数据库访问的组件
DataSnap	一组允许运算 SOAP、CORBA、COM 及 TCP/IP 等多层分布式数据库应用程序相关组件的组件
BDE	一组以新版 BDE 为基础的数据库访问组件
ADO	一组完全支持 Microsoft Universal Data Access 的 ADO 组件
InterBase	一组 InterBase 数据访问组件
WebServices	一组以 XML、SOAP 及 HTTP 为基础的 Web Service 组件
InternetExpress	一组 InternetExpress 技术的组件
Internet	开发因特网应用程序的组件

续表

组件名称	说　明
WebSnap	一组可提高 Web 应用程序开发效率的相关组件
FastNet	网络相关组件
Decision Cube	决策立体图表相关组件
Qreport	报表相关组件
Dialogs	对话框相关组件
Win31	Windows 3.1 相关组件
Samples	相关的范例组件
ActiveX	ActiveX 相关组件
COM+	COM+相关组件
Indy Clients	另一组网络客户端相关组件
Indy Servers	另一组网络服务端相关组件
Servers	一组与 Office 有关的组件

将组件添加到窗体中，最常用的方法有两种。一种方法是在组件面板上选中要添加的组件，然后在窗体上使用鼠标左键画出组件的范围，组件将自动添加到窗体上；另一种方法是在组件面板中双击要添加的组件，在窗体中央将自动添加一个固定大小的组件，然后拖曳组件周围的黑框调整组件的大小尺寸。组件添加方法如图 1.9 所示。

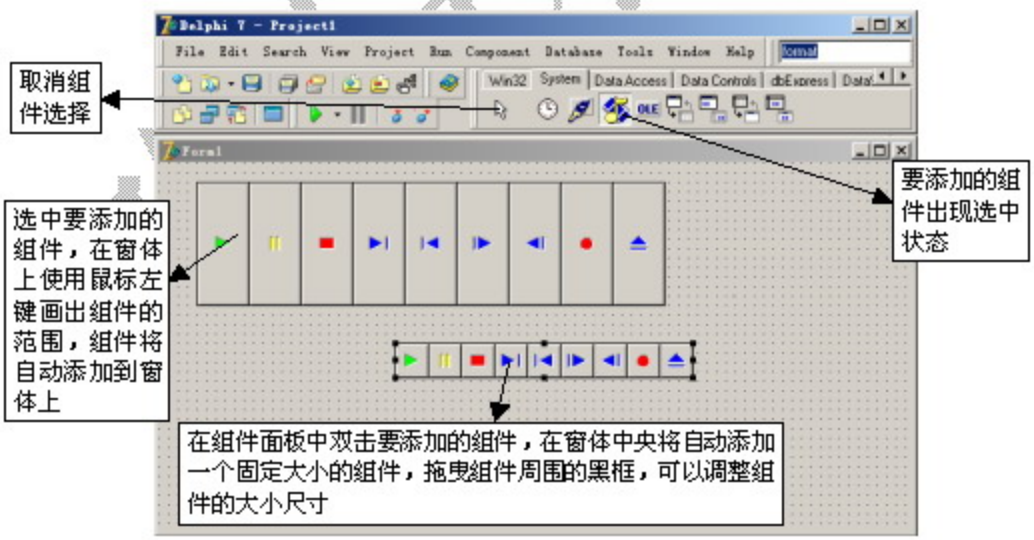


图 1.9　添加组件

如果要删除组件，可以选中窗体中的组件，按 Delete 键即可删除。

因为每个组件大小不一样，摆放的位置也有所不同，为了方便排列这些组件，Delphi 提供一个“对齐组件”的功能。选择“View”/“Alignment Palette”功能选项之后，系统会弹出一个对齐面板窗口，接着选取需要排列的组件，然后根据对齐面板上的功能来排列窗体上的组件。如图 1.10 所示。

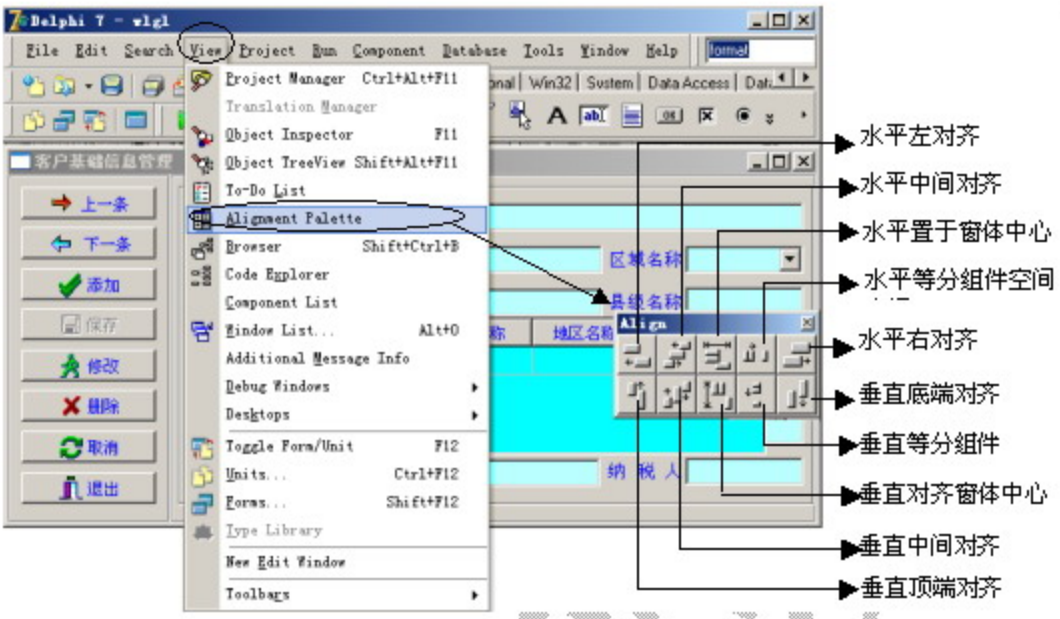


图 1.10　利用“对齐面板”设置组件对齐方式

另外，也可以通过对齐对话框设置组件对齐方式。选择“Edit”/“Align”菜单项或是选择窗体右键菜单的“Align”选项，根据弹出的对齐对话框直接设置对齐选项实现组件对齐功能，如图 1.11 所示。

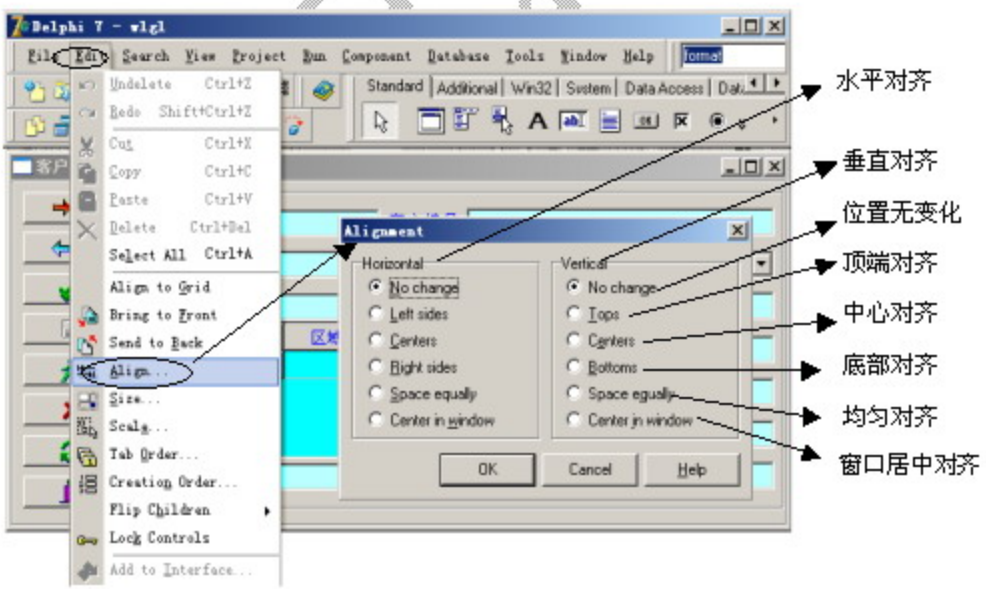


图 1.11　利用对齐对话框设置组件对齐方式

在打开的程序窗体上，画有一排一排整齐的网格（Grid），这些网格主要用于对齐窗体中的组件。选择“Tools”/“Environment Options”/“Designer”功能可以设置网格参数，如图 1.12 所示。

另外，排列好窗体上所有组件的位置之后，可以选择“Edit”/“Lock Controls”功能选项将整个窗体上所有组件的位置锁定，以防止移动鼠标时，不小心变动了组件的位置。

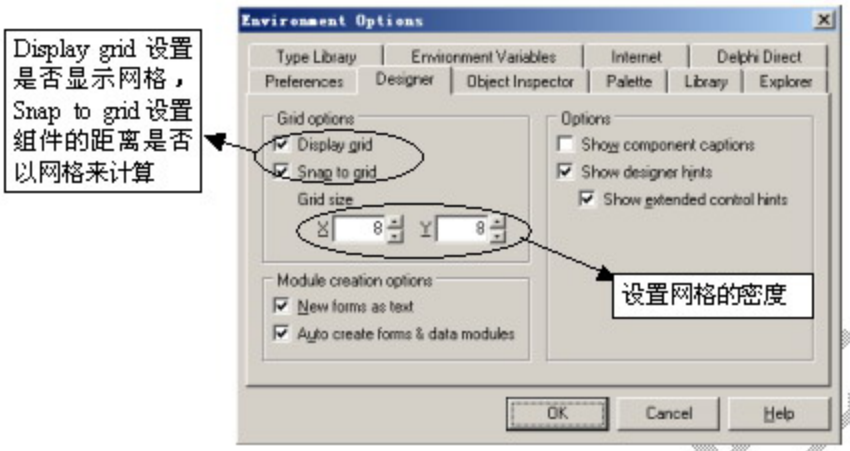


图 1.12　设置网格参数

1.2.5　对象检查器

对象检查器主要用来查看或设置窗体上的组件属性或组件的事件。如果在设计窗体时，因为打开太多窗口而遮盖住了对象检查器，可以按下（F11）功能键弹出对象检查器窗口。若要将对象检查器窗口永远保持在最上面，不会被其他的窗口覆盖住，可以在对象检查器上单击鼠标右键选择菜单中的“Stay on Top”选项，将对象检查器窗口永远保持在最上面，如图 1.13 所示。

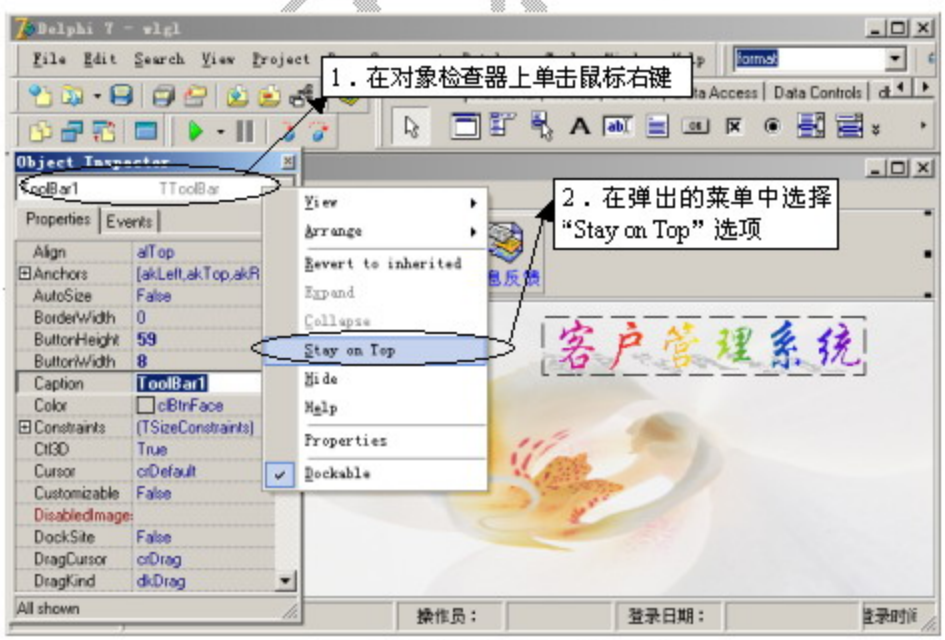


图 1.13　设置对象检查器窗口显示状态

对象检查器由对象菜单、Properties（属性）标签及 Events（事件）标签 3 部分组成，如图 1.14 所示。对象菜单提供了窗体中所有对象的菜单列表，主要为方便用户选取窗体中的某个对象。Properties（属性）用于显示或修改窗体中当前被选择部件的属性信息。属性设置可通过直接输入、列表选择（↓）或打开对话框（…）的方式选择。标记“+”表示属性含

有下层属性，双击属性名称，可展开下层属性。Events（事件）列出了当前部件可以响应的事件，用户可以通过设置事件标签中的事件函数编写响应代码。双击某一事件后边的空白处，可以定义对象接收到相应事件时执行的动作代码，如图 1.15所示。

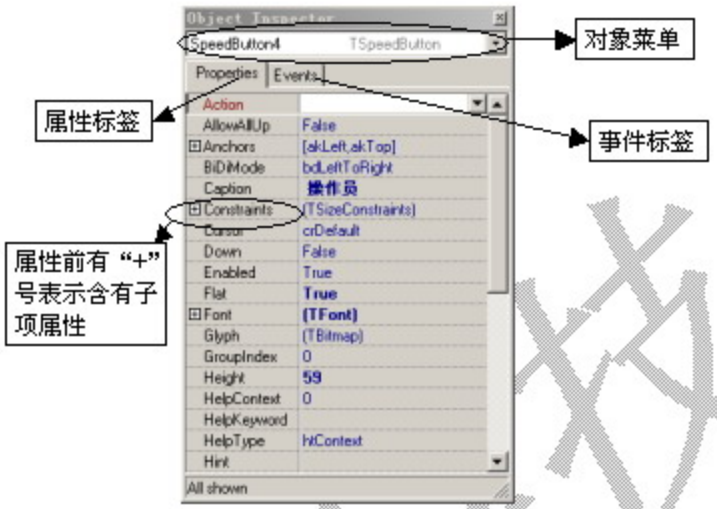


图 1.14　对象检查器

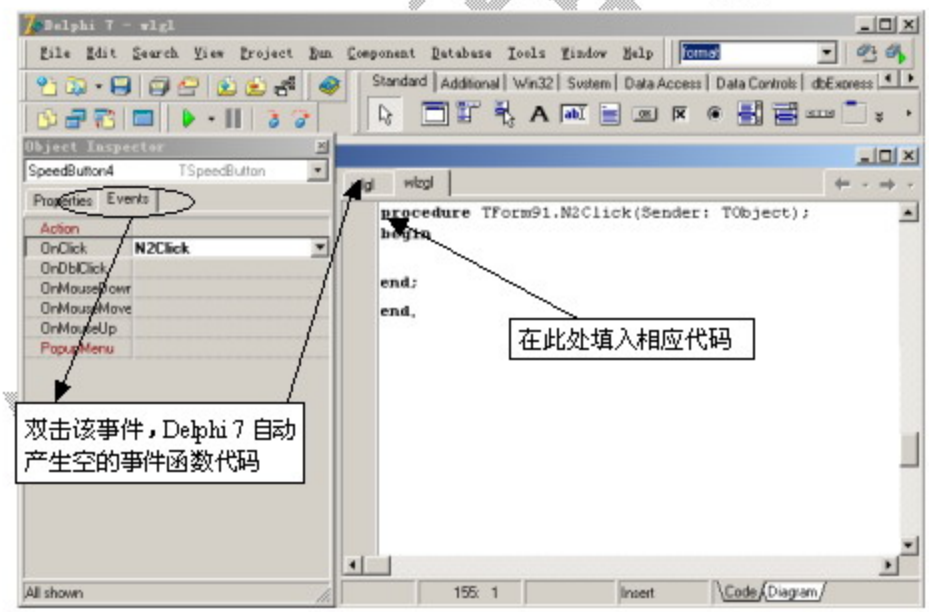


图 1.15　设置事件标签中的事件函数

1.2.6　代码编辑器

代码编辑器又称单元窗口，是编写程序代码的工具，可以同时编辑几个源程序文件，并将所有的关键字以粗体来表示，或是以不同的字体颜色来区别代表不同意义的程序块。代码编辑器其实就是一个智能的文本编辑器，它提供了一组完整的编辑功能键（如表 1.2 所示），可以让用户熟练地操作编辑功能，提高程序编写效率。许多 Delphi 专业开发人员习惯利用组合键来编辑程序内容而不是利用鼠标，这样可以大大提高程序的编写速度，所以这些组合键对于软件开发是非常重要的，应熟练掌握。

表 1.2　　编辑功能键说明

组　合　键	意　　义
Ctrl+←	将光标移到字的最左边字符
Ctrl+→	将光标移到字的最右边字符
Ctrl+Home	将光标移到程序的开头
Ctrl+End	将光标移到程序的结尾
Ctrl+K+B	设置文本块的开始
Ctrl+K+K	设置文本块的结束
Ctrl+K+R	读取某个文件插入程序中
Ctrl+K+W	将文本块的代码写入文件中
Ctrl+K+P	将文本块的代码输出到打印机
Ctrl+K+I	将文本块的代码往后缩一格空白
Ctrl+K+U	将文本块的代码往前推一格空白
Ctrl+K+C	复制文本块的代码（没有用到系统的剪贴板）
Ctrl+K+V	粘贴文本块的代码（没有用到系统的剪贴板）
Ctrl+C	复制文本块的代码到系统的剪贴板
Ctrl+V	将剪贴板中的数据复制到编辑器
Ctrl+X	复制文本块的代码到系统的剪贴板，并且删除原文本块的代码
Del	删除光标所在的字符或是文本块的代码
Ctrl+K+0,1,2,3…9	设置书签（每页程序最多 10 个）
Ctrl+Q+0,1,2,3…9	将光标移到书签处
Ctrl+F	查找字符串
Ctrl+R	查找并替换字符串

代码编辑器具有强大的代码编辑功能，主要体现在代码框架自动生成、类成员自动提示设置、参数自动提示及编辑代码中的自动保护功能。通过这些编辑功能，用户可以轻松编写 Delphi 代码。代码编辑器外观及其应用如图 1.16 所示。

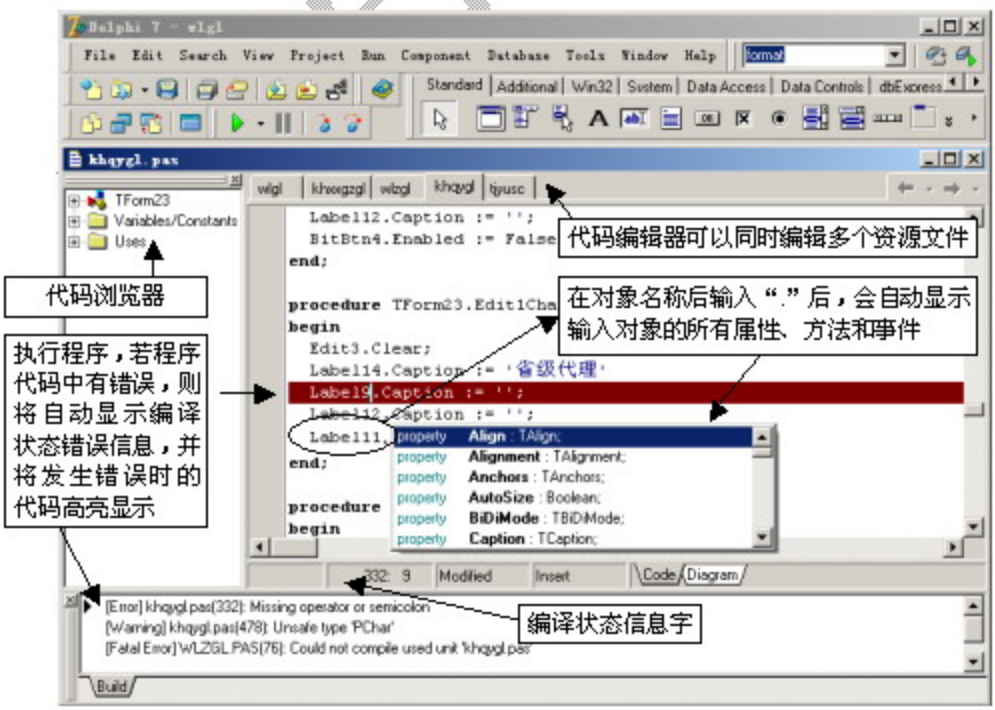


图 1.16　代码编辑器

1.2.7 代码浏览器

代码浏览器（Code Explorer）是对单元内的所有组件、属性、方法、过程、各种类型的变量进行层级树状式管理的工具。在 Code Explorer 中，可以依据不同段落的声明，展开不同的节点，往下展开一直找到所需要的节点为止，然后移动鼠标到此节点上双击，右边的 Code Editor 将会自动找到对应的声明，并且把键盘光标移到此处。利用 Code Explorer，不用借助其他工具窗口，可以很快地在 Code Editor 内找到所需要的程序段落，代码浏览器如图 1.17 所示。

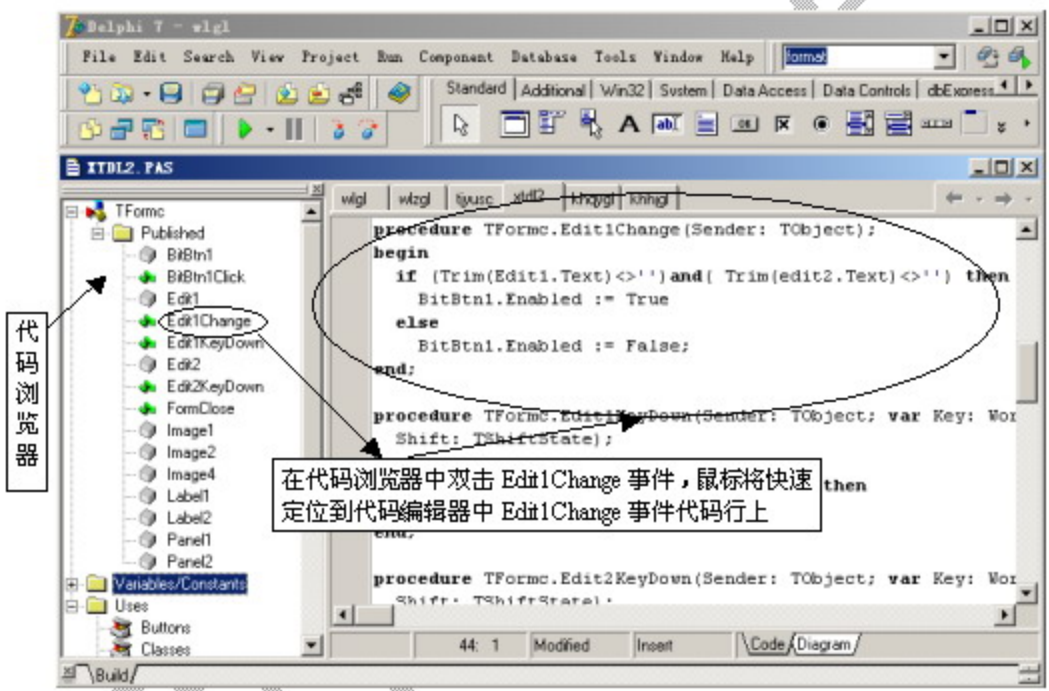


图 1.17 代码浏览器

在代码浏览器中提供了一个快速查找的功能，可以利用（Ctrl+shift+↑）键或（Ctrl+shift+↓）键快速地在函数的声明与函数的内容间快速移动。

1.2.8 对象树状窗口

对象树状窗口（Object TreeView）是一个用来显示窗体、数据模块和帧上可视或非可视组件之间逻辑关系的树状工具。当 Delphi 打开时它会出现在左上角的位置，对象树状窗口里面主要是显示对象容器（如 TForm、TDataModule、TFrame）所包含的对象（包括可视化与非可视化），并且以树状结构的方式来显示。对象树状窗口的内容会随着对象检查器（Object Inspector）与窗体设计器（Form Designer）的变动而同步改变。在实际的程序编写过程中，如果在 Form 上面包含数十个组件，而要在这数十个组件中找到所需要的组件，会变得很麻烦，对象树状窗口则会把这些组件创建成树状结构，通过它很轻易地就能找到想要点选的组件，对象树状窗口如图 1.18 所示。

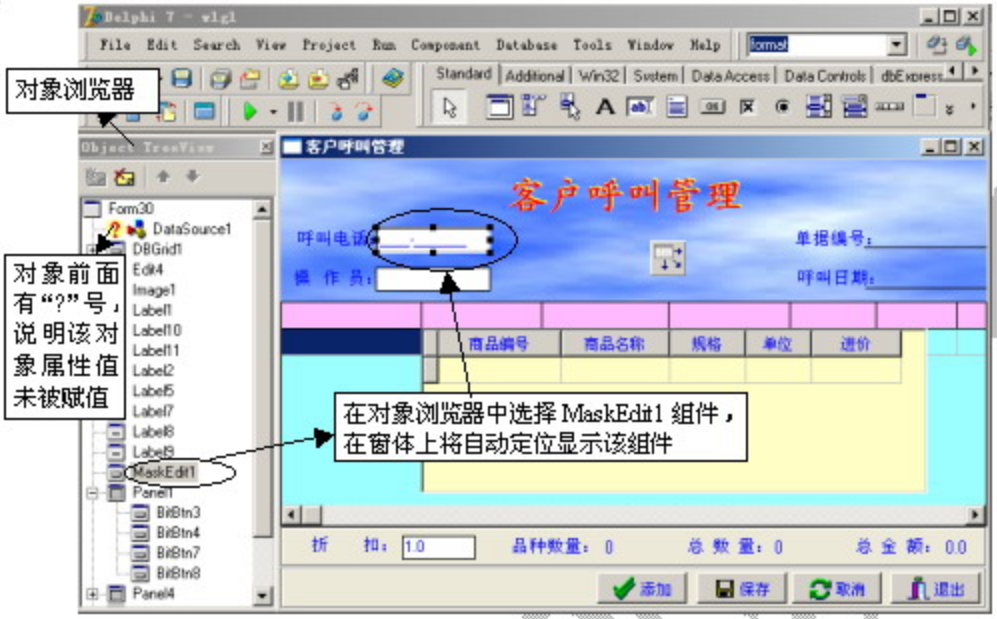


图 1.18　对象树状窗口

1.2.9　项目管理器

项目管理器的主要功能是集中管理应用程序所包含的所有窗体文件与代码单元。在应用程序的开发中，一般每个窗体都可作为一个部分单独进行设计，然后利用项目管理器将这些窗体组合起来，项目管理器如图 1.19 所示。

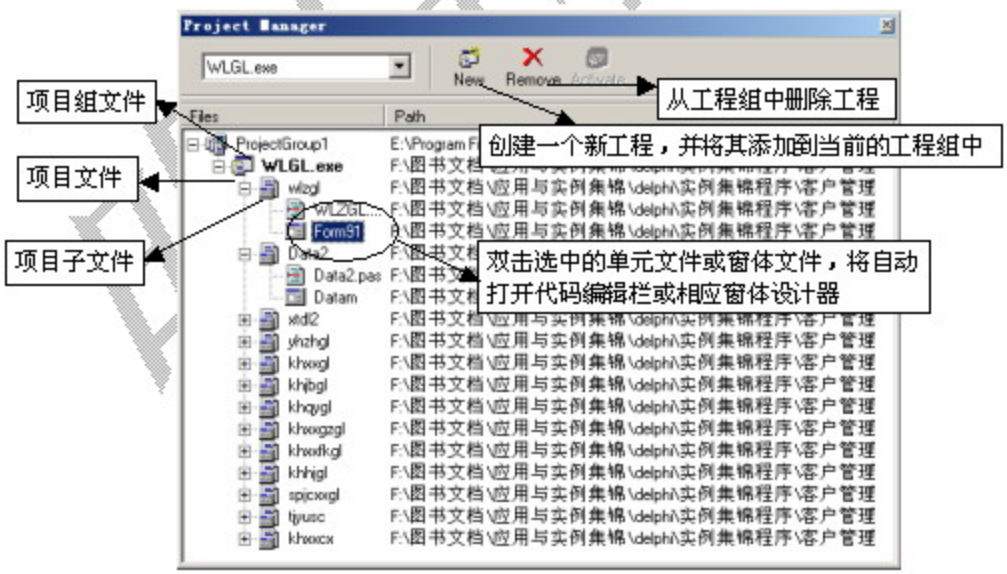


图 1.19　项目管理器

通过项目管理器可以改变项目中的成员组成，但对这些成员还要设定另外一些信息，这可以通过进入“Project”/“Options”菜单项的 Forms 选项卡来进行设定。其操作界面如图 1.20 所示。在该选项卡中可以指定当前项目中的主窗体，并指定各窗体的生成方式。

在一个项目中必定要有一个主窗体，这个主窗体由系统在项目文件中加入相应的语句

自动生成，也就是当该项目被启动时，首先显示主窗体。对于其余的窗体其生成的方式也有两种。一种是像主窗体那样，由系统在项目文件中加入相应的语句自动生成。采用这种方式的窗体在项目启动时就全部被建立，虽然编码是自动生成的，但是它们都过早地占用了内存资源。另一种是引用时才被打开，也就是设计者在程序中要用到某窗体时，使用该窗体的 Create 方法来建立。采用这种方式的窗体在引用时被建立，在不使用时被释放，因此比较节省内存资源，但必须由设计者进行编码。这两种方式在使用上各有特点，设计者可以根据设计的要求进行选择。

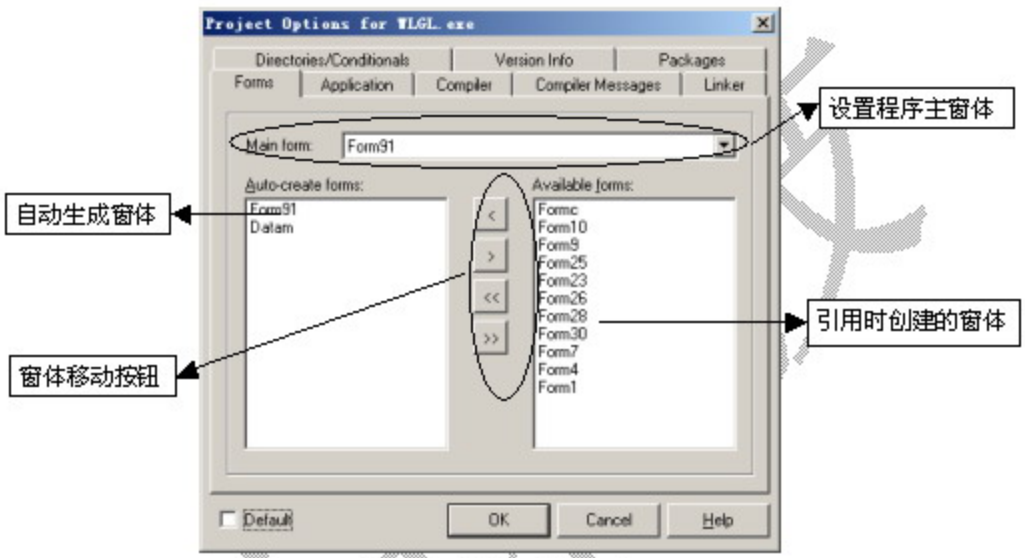


图 1.20　窗体选项的设置

设置这两种方式的操作过程相当简单。在该选项卡中有“Auto-create forms”以及“Available forms”两个小窗口，在这两个小窗口之间有 4 个移动按钮。如果将某窗体的名称移向“Auto-create forms”一边，则该窗体为自动生成窗体；如果将某窗体的名称移向“Available forms”一边，则该窗体为引用时生成窗体。

1.3　应用程序的开发过程

熟悉了 Delphi 7 的开发环境后，接下来将通过一个简单的应用程序的设计过程介绍应用程序的开发过程和应用程序的基本结构。

1.3.1　制作一个简单的应用程序

本小节要创建一个简单的应用程序。当程序被启动时，在窗体的标题栏中显示“捷锐 EMS 快运系统”和窗体图标，在窗体中显示立体阴影文字“捷锐 EMS 快运系统”，在窗体下显示文字“明日科技有限责任公司”，在窗体右下脚显示一个“退出”按钮。当单击“退出”按钮时程序退出执行。其界面外观如图 1.21 所示。

为了创建上述程序并尽快地熟悉 Delphi 的开发环境与开发过程，可按下面步骤执行有关的操作。



图 1.21　一个简单的应用程序

1. 新建工程

启动 Delphi，进入 Delphi 的开发环境。单击“File”/“New”/“Application”菜单项，新建应用程序，如图 1.22 所示。

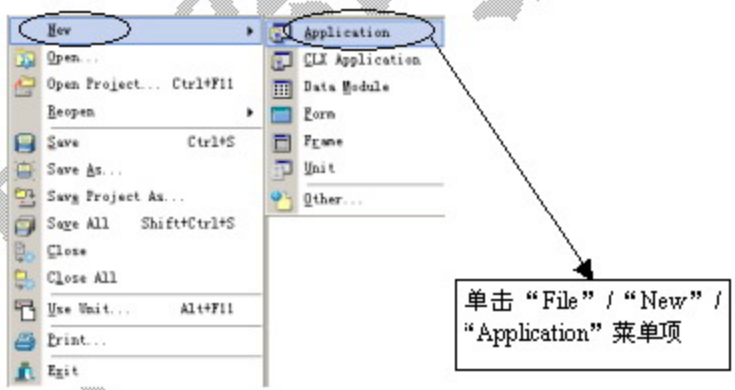


图 1.22　新建应用程序

2. 设置窗体的属性

首先设计窗体的标题栏（Caption）和窗体图标。在对象监视器中选择 Caption 属性，设置属性值栏中的值为“捷锐 EMS 快运系统”后按 Enter 键。这时窗体标题栏中的显示文字都已改成了“捷锐 EMS 快运系统”。在对象监视器中选择 Icon 属性，在图标编辑栏中选择合适的图标，为窗体设置窗体图标，如图 1.23 所示。

接下来调整窗体的大小。把鼠标移到窗体的边沿，拖曳鼠标直接改变整个窗体的大小，指定代码单元文件名为 Main，则窗体名为 mainForm。

3. 设置立体阴影字“捷锐 EMS 快运系统”

立体阴影字“捷锐 EMS 快运系统”可以通过两个不同颜色的 TLabel 对象错位叠加实现。首先选中组件面板的“Standard”标签，从中选取 TLabel 组件，在窗体中拖曳鼠标将 Label 组件添加到所选位置，指定其 Caption 属性为“捷锐 EMS 快运系统”，设置 Transparent 为 True，使其透明。接下来调整 TLabel 组件的字体、样式、大小及颜色。在对象观察器上

单击 Font 属性值右边的小按钮，弹出“字体”对话框。Label1 组件字体属性设置如图 1.24 所示。

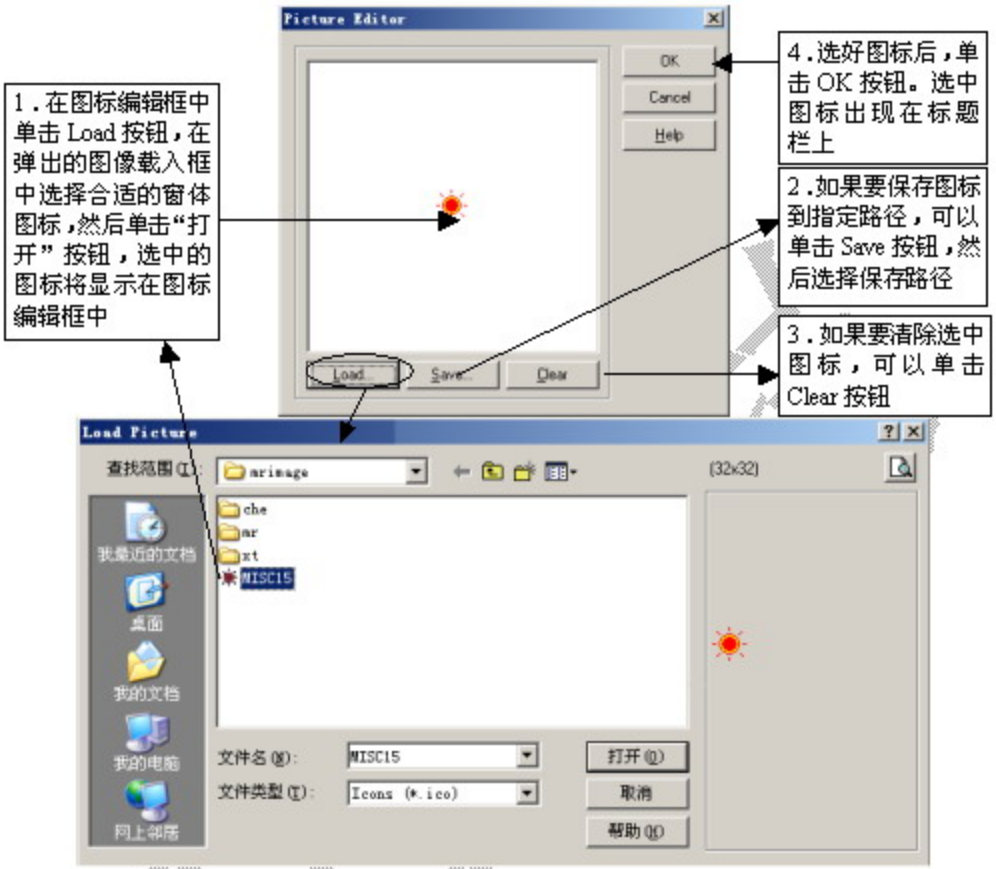


图 1.23　为窗体添加图标



图 1.24　“字体”对话框

单击 Label1 组件，选择“Edit”/“Copy”菜单项复制 Label1 组件。然后单击窗体，选择“Edit”/“Paste”菜单项粘贴 Label1 组件，其 Name 属性默认为 Label2，设置其字体颜色为大红。并调整 Label2 的位置，使其与 Tlabel 对象错位叠加，产生立体阴影字效果。

4. 设置开发单位和“退出”按钮

在窗体下部添加一个 Tlabel 组件，用于显示软件开发单位名称。设置其 Caption 属性值为“明日科技有限责任公司”，使用鼠标将其拖曳到立体阴影字右下方。

在窗体的下部设置按钮组件 Button1，用于执行窗体的关闭操作。指定其 Caption 属性为“退出”，使用鼠标将其拖动到窗体中的适当位置，并拖动它的边沿将其调整到适当大小。

5. 设置按钮事件处理

为了使“退出”按钮能够在执行时响应鼠标的单击操作，需在按钮 Button1 的单击事件（OnClick）中设置事件处理程序。选择 Button1 对象，在对象监视器中选择 OnClick 事件，双击事件处理栏进入代码编辑窗口，也可以直接双击 Button1 对象进入代码编辑窗口，系统自动生成过程的框架如下：

```
procedure TExamForm.Button1Click(Sender:TObject);
begin
{光标位置}
end;
```

在过程体中可设置如下的代码：

```
close;
```

执行该代码时即可将当前的窗体关闭。

在本应用程序的整个开发过程中，惟一涉及编程代码的就是这一个 close 语句，其余的都可以通过设定、选择等一些简单的操作来完成。这也充分体现了 Delphi 开发工具强大的可视化的设计功能。

6. 保存文件

在 Delphi 的开发环境中新建一个项目时，系统便自动生成了一个新的项目，包括 3 个文件，这些文件的名称，默认文件名及后缀名，如下：

- 项目文件　Project1.dpr
- 窗体文件　Form1.dfm
- 单元文件　Unit1.pas

作为一个实际的应用程序，应该将这些默认的文件名更改为更有意义、更便于记忆、不容易混淆的文件名。特别是当项目中包含的文件较多时，如果文件名称指定不当，会给项目的开发带来意想不到的麻烦。一般单元文件名和窗体文件名可取同一个文件名，由后缀名为它们进行区分，窗体名可以取与窗体文件名相关的名称。例如，将窗体文件名确定为 EMS，其他名称可确定如下：

- 项目文件　Emsprj.dpr
- 窗体文件　Ems.dfm
- 单元文件　Ems.pas
- 窗体名称　EmsForm

文件名及窗体名确定以后，就应当将它们保存起来，以防止异常的运行结果使前面已开发的成果毁于一旦。

对于窗体文件和单元文件可通过“Save As”菜单项来取名保存，对于项目文件可通过“Save Project As”菜单项来取名保存，对于窗体名称可通过指定该窗体的“Name”属性来

设置，全部取名保存以后，下次再保存时可使用 Save All 菜单项来保存项目中所包含的全部文件。

7. 运行程序

运行程序有两种方法，一是选择“Project”/“Compile”菜单项对项目中的文件进行编译，如果语法有错，则根据提示的错误信息修改后再编译。另一种方法是选择“Run”/“Run”菜单项（或单击 Run 快捷按钮，或按 F9 键）进行编译与运行。

1.3.2 构成 Delphi 7 项目的主要文件

在 1.3.1 小节中设计了一个简单的应用程序，下面打开资源管理器（如图 1.25 所示），看看都生成了哪些文件。

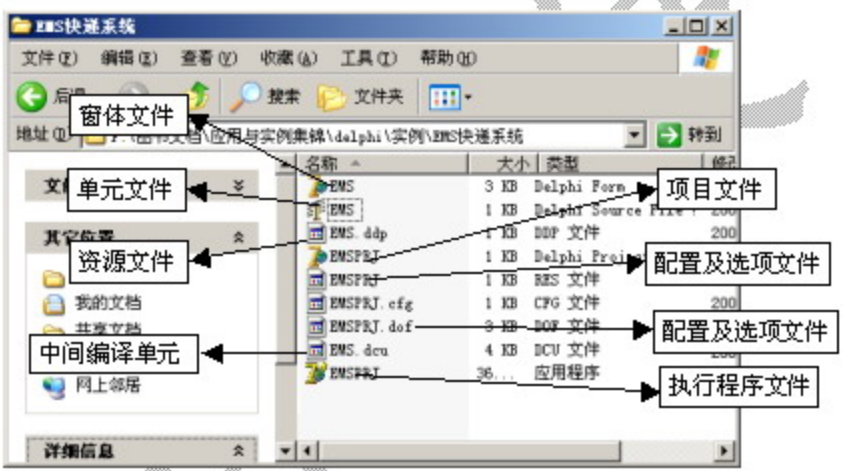


图 1.25 资源管理器

在资源管理器中，主要生成了下面一些文件。

- EMSPRJ.dpr 项目文件
- EMS.dfm 窗体文件
- EMS.pas 单元文件
- EMSPRJ.exe 执行文件

项目文件（后缀名为 dpr）：可选择“View”/“Units”菜单项，进入“ViewUnit”对话框。在该对话框中列出了该项目中所有的单元文件名，也列出了项目文件的文件名。选择项目文件后单击 OK 按钮，即可查看该项目的项目文件。

单元文件（后缀名为 pas）：查看单元文件也可选择“View”/“Units”菜单项。

窗体文件（后缀名为 dfm）：选择窗体的右击弹出菜单中的“View as text”菜单项，即可查看到窗体的内容。

执行文件（后缀名为 exe）：这个文件为执行文件，在编译时产生，所占的存储空间比较大，而且是通过运行项目来生成的，所以在做成应用程序的备份文件时不一定要包括它，但可以从资源管理器中双击“EmsPrj.exe”文件直接运行应用程序。

1.3.3 构成 Delphi 7 项目的主要文件说明

- 项目文件（.dpr）：是在设计时创建的，为主程序文件。是主窗体以及其他自动创

建的窗体实例化的地方，一般不需要编辑项目文件，除非要执行程序初始化例程、显示启动画面或执行其他必须在程序启动时执行的例程。下面代码是一个典型的项目文件。

```
program Project1;
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.res}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Pascal 程序员会把项目文件看做是标准的 Pascal 源程序。

{$R *.res}：用于引用项目的资源文件（扩展名为 res）。

begin…end 之间的语句：是应用程序的执行代码。主要用到了 Application 的 3 个方法。Initialize 方法是初始化，CreateForm 方法是建立窗体，Run 方法是运行程序。

- 窗体文件（.dfm）：它存储了窗体中所有对象的设计属性。以二进制形式保存。
- 单元文件（.pas）：与一个窗体文件相对应，在单元文件中定义了某一个窗体对象（类）。但也有的单元文件中定义了其他一些类，或者只定义了一些公用的过程或函数，并不是与窗体的定义相对应。当用户新建一个项目或新添加一个窗体时，系统会自动生成一个相对应的代码单元。

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs;
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
begin
end;
end.
```

注意：向窗体中添加组件对象时，如果该组件对应的类不在 user 所应用的单元中，系统会自动加上所需单元。如果要使用属于其他单元的过程或函数，程序员应在 user 中加上相应的代码单元名。

- 资源文件（.bas）：包含了二进制数据，也称为资源，这些资源将链接到应用程序的可执行文件中，包含应用程序的图标、版本信息及其他信息。
- 项目选择及桌面设置文件（.dof）：存储了“Project”/“Option”选项卡所设置的项目选项，它在第一次保存项目时创建，以后每次保存项目都会保存这个文件。

明日科技

第 2 章 面向对象的应用程序设计

2.1 面向对象的程序设计

2.1.1 对象的引入

对象是在面向对象的程序设计中提出的。在面向对象技术出现以前，开发人员都是采用传统的结构化分析方法，将整个系统视为一个大的处理过程，然后将其进行分解成为若干个子处理过程，在程序中用数据描述过程间的联系。这种设计方法不灵活，尤其是在需求分析有变化时，程序需要进行大量的修改。在这种情况下，人们提出了面向对象的设计思想，即以对象模型为单位，将系统功能的实现细节封装在对象内部，外界只能根据接口访问对象中的相关数据。

对象是客观事物在人脑中的映像，人脑通过对这种映像进行抽象进而形成概念。例如，现实生活中的人、汽车、电脑是对象，人有白皮肤的、黄皮肤的、黑皮肤的，能够说话、走路，能够劳动、生产，这样的事物是“人”。当我们一提到“人”时，便在大脑中形成了“人”的概念。

在面向对象程序设计中，对象是以数据结构的方式存储在计算机中的，数据结构中包含了对象的属性和行为。属性是一个对象区别于其他对象的标识。例如，人有年龄、姓名、性别、肤色、国籍等属性，而每个人都有自己的年龄、姓名、性别等属性值。对象的行为也称之为方法，是指对象能够做的事情。例如，人能够说话、走路、劳动，飞机能够飞行和降落。

基本上，软件对象本身具有 3 项特性，分别是继承、封装、动态联编。

“继承”特性在面向对象设计上，是非常重要的一项概念，同样的数据和代码，不必在每个需要的地方再重写一遍，只要归纳整理成一个类，需要时再重制出所需要的实例即可。简单地说，继承性是类之

间的一种再分类关系，它会让几个类之间具有相同的数据结构或操作过程，减少重复的代码，达到对象重复使用的目的。

“封装性”的意思是把整个对象的数据和处理过程“包”在一起，通过一个标准、简单的接口才能访问对象内的数据及使用对象提供的功能，这所谓的接口就是指对象的属性（property）和方法（method）。由于软件的开发功能越来越强，相对地也使得系统变得更复杂，为了降低整个系统的复杂度，所以就把整体分成许多块对象，将每一块对象的复杂度降低到一般软件开发者可以负荷的程度，而且让开发者能针对每一块对象去做分析、设计的工作，不会影响到其他块对象，这就是对象必须具有的独立特性，而对象封装的目的就是为了达到对象的独立。但是，对象封装的前提是对象里面封装的代码必须是正确的，而且要符合当初设计的需求。

“动态联编”是指在程序运行时，系统才将 message 和 method 联编在一起，而不是在编译的时候。message 是送达对象的接口，用来启动对象的工具，而 method 则是在 message 送达对象接口后，被选取的操作过程。例如，爸爸叫儿子去买早点，爸爸对象只需要把“买早点”的 message 送给儿子这个对象，而不需要知道儿子对象是如何应用 method 来完成“买早点”的操作，若是儿子有好几个，动态联编会自动把 message 送给适合的对象 method 运行，无须担心是哪个儿子对象完成的。因此，通过动态联编的功能，我们不用事先考虑使用对象的类型，只要把 message 送出即可，等到程序运行时，看哪一个对象应该配合这个 message，再把 message 和这个对象的 method 联编，然后运行这个 method。

2.1.2 Delphi 提供的对象

Delphi 提供一组 VCL（Visual Component Library）组件，这些组件的数量超过 200 个，由于组件的数量太多，而且未来程序设计师也有可能自行安装其他的组件，所以 Delphi 就将这些组件做分类处理，方便程序设计师快速地取用这些组件，不会像其他的可视化开发工具，每次找到组件都要花费很长时间。而且 Delphi 将这些分类过的组件，采用分页的显示方式摆置，所以只占屏幕一小块位置，节省宝贵的屏幕空间。对于 Windows 的程序设计师而言，永远嫌屏幕太小，摆不下所有用到的窗口，用户只需要单击组件面板右端的 3 个箭头，就可以选用其他标签的组件。

Delphi 将这组 VCL 组件分成两种模式，一种是可视化的组件，另一种是非可视化的组件。所谓可视化的组件是指设计时期放置在 Form 上的组件外观和运行后的组件外观是一致的，可以直接设置组件的属性并加上事件处理过程，对象的外观将会随着属性的改变而有所不同，如 Standard、Win32、Additional、Data Controls 标签内的组件；而非可视化的组件在设计时期可以看到小图标出现在 Form 上，也可以直接设置其属性及加上事件处理过程，但是不能改变外观，而且运行后会看不到组件的样子，可是组件仍在运行，如 Data Access、Dialogs 标签内的组件。在图 2.1 所示的窗体中，①表示的组件为非可视化组件，其大小固定不可改变；②表示的组件为可视化组件。

整个 VCL 组件库内的所有组件都是利用继承特性产生的，所以这些组件可能因为继承的对象不同，而有不同的功能及特性。例如，继承自 TButtonControl 对象的 TButton、TBitBtn、TRadioButton 等对象，都具有命令按钮的特性；而继承自 TGraphicControl 对象的 TBevel、TImage、TShape 等对象，都具有图形的特性功能。基本上，这些对象的继承关系，可创建成一幅树状关系图，从这张图中可以发现，整个 VCL 所包含的对象功能特性，都是根据上层的几个基本对象而定的，如一些非可视化组件大部分都是继承自 TComponent 对象，而

一些可视化组件大部分都是继承自 TControl。在程序设计时，如果希望直接继承 VCL 内的组件来编写自己的组件，那么这份 VCL 的继承关系图将会非常重要。



图 2.1　应用程序中的窗体

2.2　认识 Delphi 的窗体

在 Windows 操作系统中，人机交互主要是通过一些窗口和对话框实现的。在 Delphi 中，窗口就是程序设计阶段的窗体。窗体是创建 GUI（窗口程序）程序的基本元素，它是 Delphi 面向对象程序设计最主要的对象，几乎所有的应用程序都是以窗体对象为基础进行程序开发的。它直接封装了 Windows 桌面上的可见窗口，起着一个容器的作用。可以说窗体组件是应用程序的核心，如果没有窗体组件，一切图形用户界面就不存在了。一个应用程序通常包含多个窗体，其中有一个是主窗体，其余窗体是应用程序窗体和对话框窗体。在图 2.2 中，说明①表示的是程序主窗体，说明②表示的是应用程序窗体，说明③表示的是对话框窗体。主程序是应用程序的主要用户接口。用户程序窗体主要是由主程序窗体调用的。

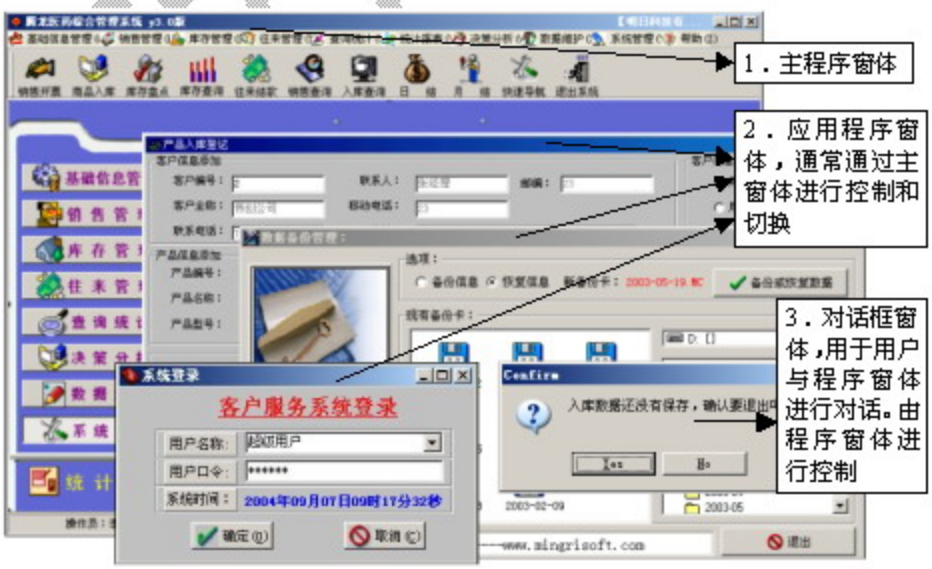


图 2.2　应用程序中的窗体

在设计期间，窗口通常叫窗体。在程序执行期间，一般叫做窗口。窗体本身也是对象，所以对象有的属性和方法，窗体也都有。通过这些属性和方法可以设计符合要求的应用程序窗体。可以直接在对象检查器上修改属性，或是在程序中设置属性及调用方法。

从用户角度来说，窗口是实现部分功能的屏幕区域，它一般有标题栏、系统菜单、“最大化”、“最小化”、“关闭”按钮。可以通过拖曳鼠标使窗口在屏幕上移动，也可以最大化、最小化、还原或关闭程序窗口。

2.3 窗体对象的主要属性

2.3.1 与窗体位置、大小有关的属性

窗体对象向外部提供了 4 个属性用来设置其位置与窗体大小，这些属性在设计窗体界面时会经常用到，窗体对象属性如图 2.3 所示。

- Left：该属性确定窗体的左边界。
- Top：该属性确定窗体的上边界。
- Height：该属性确定窗体的高度。
- Width：该属性确定窗体的宽度。



图 2.3 与窗体位置、大小有关的属性

2.3.2 与窗体客户区大小有关的属性

在 Windows 窗口中，窗体大小包括标题栏、菜单栏和边框的窗体大小；客户区的大小不包括标题栏、菜单栏和边框，但包括工具栏和状态栏。默认情况下，窗体边框的宽度为 4 像素，标题栏的高度为 19 像素，菜单栏的高度为 20 像素。如在“工资管理系统”中，工资管理主窗口大小为 800 像素×600 像素，那么客户区的宽度应为窗体宽度减去左右边框的宽度，即 800-2×4=792 像素，客户区的高度应为客户区的高度减去上下边框高度和标题栏、菜单栏的高度，即 600-2×4-19-20=553 像素，如图 2.4 所示。

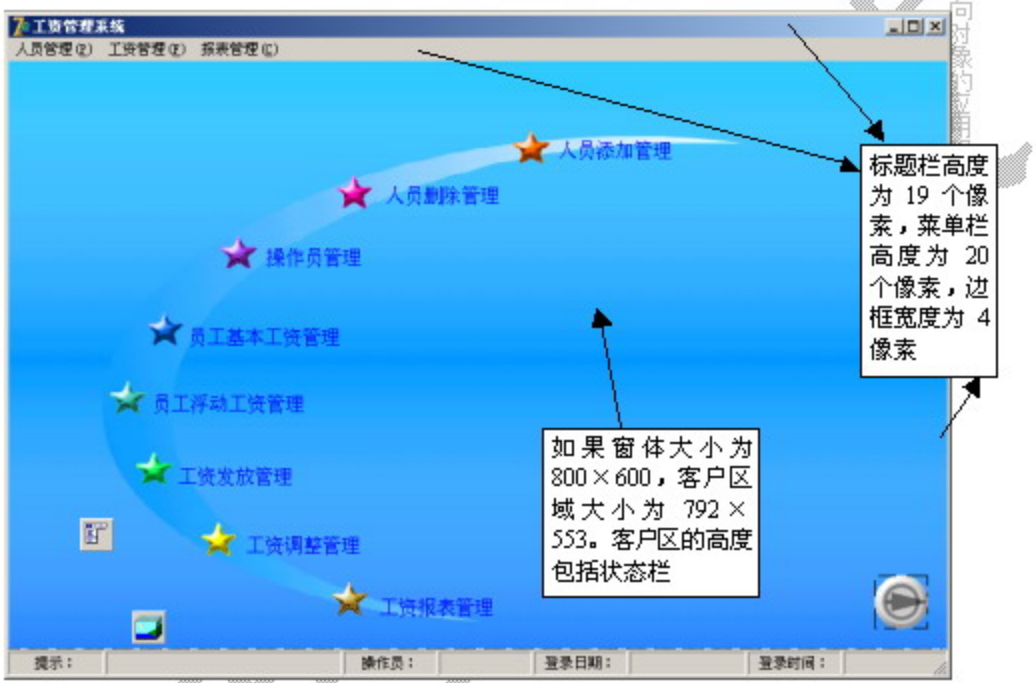


图 2.4 窗体客户区大小

与窗体客户区大小有关的属性如下。

- ClientHeight：窗口客户区的高度。
- ClientWidth：窗口客户区的宽度。

Height 与 Width 属性确定了窗体的高度和宽度，ClientHeight 与 ClientWidth 属性则确定了窗口客户区的高度和宽度。在编写应用程序时，应注意到客户区域与窗体大小是否一致，设计界面和窗体布局应充分考虑这些变化。

2.3.3 与窗体显示有关的属性

与窗体显示有关的属性主要有 Visible、Color 和 Font，这些属性主要设置窗体的可见性、窗体背景颜色及窗体字体大小，窗体对象与显示有关的属性设置如图 2.5 所示。

- Visible：该属性确定窗体是否可见。
- Color：该属性确定窗体的颜色。
- Font：该属性确定窗体的字体。

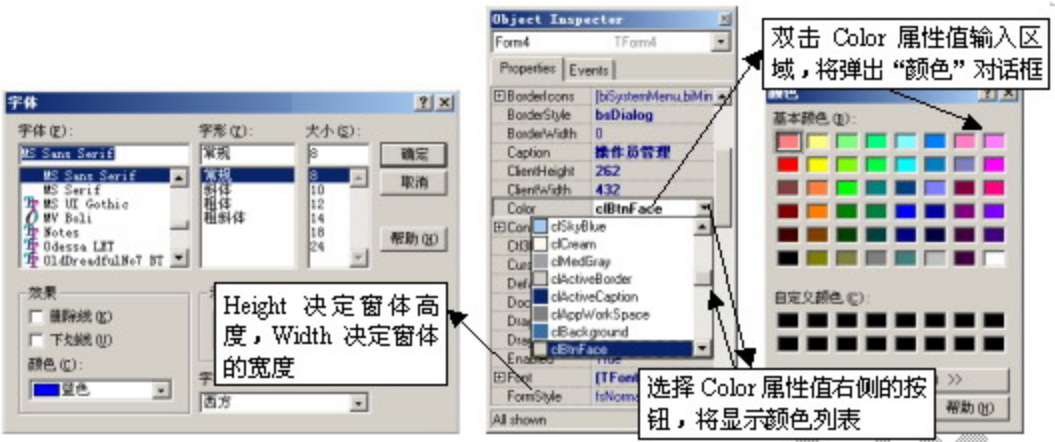


图 2.5　与窗体显示有关的属性设置

2.3.4　与窗体透明有关的属性

在 Delphi 中可以轻松制作半透明的窗体。Delphi 提供了 AlphaBlend、AlphaBlendValue 和 TransparentColor、TransparentColorValue 两类与窗体透明有关的属性。这两类属性设置在设计时期是看不到效果的，需要执行程序才能看到设置的效果。

- AlphaBlend：设置窗体是否透明（适用于 Windows 2000 以上系统）。
- AlphaBlendValue：设置窗体半透明程度（0～255），数值越小，透明程度越高。
- TransparentColor：设置窗体是否使用透明色。
- TransparentColorValue：设置窗体上预透明的颜色值。

在“客户管理系统”中设置客户管理系统主界面窗体透明的步骤如下。

（1）启动 Delphi 7，打开客户管理系统。选择主程序窗体。

（2）按 F11 功能键，打开对象检查器（Object Inspector）窗口。

（3）设置窗体对象的 AlphaBlend 属性值为 True，AlphaBlendValue 属性值为 200。运行程序，效果如图 2.6 所示。

（4）如果想要去掉主窗体中某种颜色值，可以设置窗体对象的 TransparentColor 属性值为 True，TransparentColorValue 属性值为需要透明的颜色值，如 clGreen（绿色）。运行程序，可以看到窗体中的绿色区域已变成完全透明。

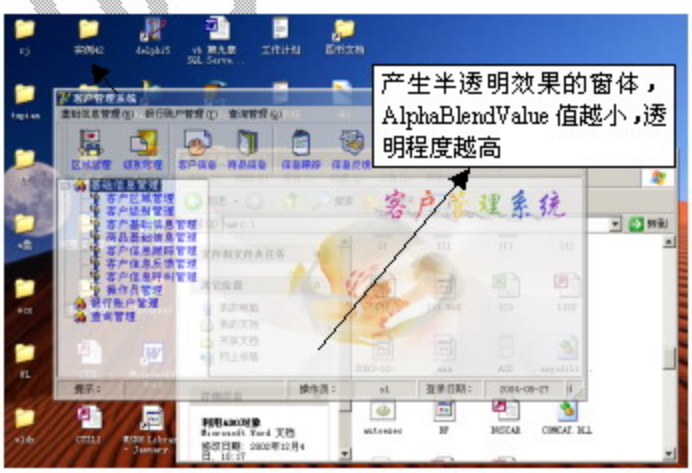


图 2.6　设置窗体透明

2.3.5 与窗体外观有关的属性

与窗体外观有关的属性主要有 WindowState、FormStyle、BorderIcons 和 Caption，这些属性主要设置窗体的状态、边框、标题栏按钮及窗体标题文字。相关属性详细介绍如下。

- WindowState：该属性确定窗体初始化时的显示状态，取值为“wsNormal”、“wsMaximized”或“wsMinimized”。“wsNormal”表示正常状态，即窗体会按照设计时的大小显示。“wsMaximized”表示运行时窗体会最大化显示，“wsMinimized”表示运行时窗体会最小化显示。
- FormStyle：该属性确定窗体的风格，取值为“fsNormal”、“fsMDIChild”、“fsMDIForm”或“fsStayOnTop”。“fsNormal”表示窗体是普通窗体，即不是多文档父窗体，也不是多文档子窗体。“fsMDIChild”表示窗体是多文档子窗体。“fsMDIForm”表示窗体是多文档父窗体。“fsStayOnTop”表示窗体将始终处于桌面的上方。FormStyle 属性设置如图 2.7 所示。
- BorderIcons：该属性确定窗体标题栏显示的按钮。取值为“biSystemMenu”、“biMinimize”、“biMaximize”或“biHelp”。“biSystemMenu”表示包含“系统菜单”，即窗体标题栏最左端的下拉菜单。“biMinimize”表示窗体只包含“最小化”按钮。“biMaximize”表示窗体只包含“最大化”按钮，如图 2.7 所示。
- Caption：该属性确定窗体标题栏显示的标题文字。

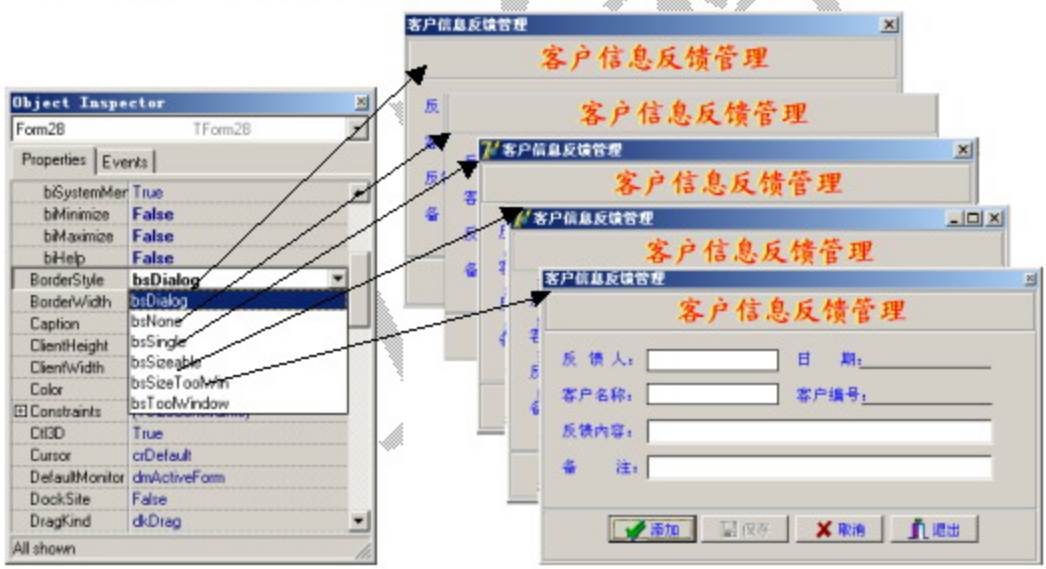


图 2.7 FormStyle 属性设置

2.3.6 窗体的绘图属性

Delphi 的窗体对象有一个画布属性“Canvas”，通过调用该属性的相关方法便可以在窗体中绘制各种图形。“Canvas”属性实际上是一个 TCanvas 类型对象，它通过窗体对象的“GetCanvas”方法指向一个 TCanvas 类型对象。下面介绍 TCanvas 类的常用属性和方法。

- Arc

该方法可以在界面上画一个弧，原型如下：

```
procedure Arc(X1,Y1,X2,Y2,X3,Y3,X4,Y4: double);
```

说明：参数 X1 表示弧边界矩形起始点 x 轴坐标，Y1 表示弧边界矩形起始点 y 轴坐标；X2 表示弧边界矩形终止点 x 轴坐标，Y2 表示弧边界矩形终止点 y 轴坐标；X3 表示弧的起始点 x 轴坐标，Y3 表示弧的起始点 y 轴坐标；X4 表示弧的终止点 x 轴坐标，Y4 表示弧的终止点 y 轴坐标。

- Brush

该属性确定用于绘画填充的颜色和图案。

- CopyRect

该方法将一块画布上的一个矩形区域复制到另一块画布上，原型如下：

```
procedure CopyRect(const Dest: TRect; Canvas: TCanvas; const Source: TRect);
```

说明：参数“Dest”表示目的矩形区域，“Canvas”表示原画布，“Source”表示原矩形区域。

- Draw

该方法在画布上画一幅指定的图形，原型如下：

```
procedure Draw(X, Y: Integer; Graphic: TGraphic);
```

说明：参数“X”表示绘制位置的 x 坐标，“Y”表示绘制位置的 y 坐标，“Graphic”表示要画到画布上的位图、图标或图元文件。

- DrawFocusRect

该方法绘制一个焦点矩形，原型如下：

```
procedure DrawFocusRect(const Rect: TRect);
```

说明：参数“Rect”表示焦点矩形。

2.3.7 与窗体控制有关的属性

与窗体控制有关的属性主要有 ComponentCount、Components、ControlCount 和 Controls，相关属性介绍如下。

- ComponentCount

该属性是指窗体对象所拥有的组件数，换句话说，窗体中有多少个组件的“Owner”属性是窗体对象。

- Components

该属性指定窗体对象上的控件数组。通过数组下标可以访问窗体所拥有的对象。

- ControlCount

该属性指定有多少个控件的“Parent”属性是窗体对象。

- Controls

该属性是“Parent”属性为窗体对象的控件的数组。

2.3.8 与窗体约束有关的属性

在设计程序时，如果不考虑窗体的约束性，常常会使窗体界面变得很难看或干脆不能使用。如图 2.8 所示，“人员添加”窗体下面被用户不小心调高了，窗体显得非常难看。“工资调整”窗体被用户将宽度调小了，窗体上的一些组件和按钮无法显示，这样会给用户

带来很大的不便。如何避免这样的情况发生呢？Delphi 为窗体组件提供了 Constraints 属性，此属性可以限定窗体的最大与最小值，我们称其为窗体约束。只需要将 Constraints 属性的各个子属性设定为特定值，即可将窗体的大小限定在特定的范围内。Constraints 属性的各个子属性说明如下。

- MaxHeight：该属性是指窗体对象可调整的最大高度。
- MaxWidth：该属性是指窗体对象可调整的最大宽度。
- MinHeigth：该属性是指窗体对象可调整的最小高度。
- MinWidth：该属性是指窗体对象可调整的最小宽度。

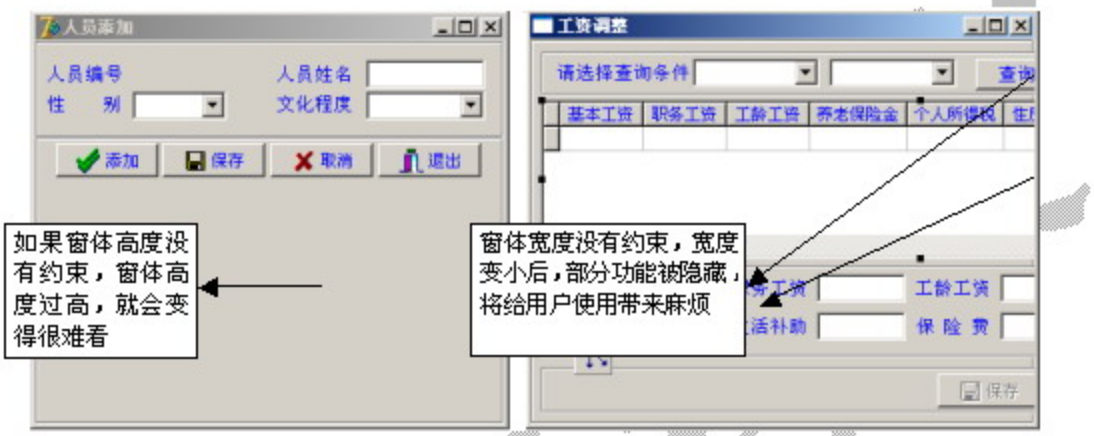


图 2.8　没有约束的窗体

2.3.9　与窗体字体比例调整有关的属性

在设计程序主窗体界面时，常常设计主窗体能自动适应用户界面，也就是说，不管用户屏幕分辨率是 800 像素×600 像素还是 1024 像素×768 像素，窗体和组件的大小都要自动根据用户屏幕调整程序主界面的大小。但出现了一个问题，随着窗体、组件显示比例与用户大小的调整，窗体上的字体没有相应变化，影响了窗体的整体效果。该如何调整窗体中字体的大小呢？下面介绍的几个属性可以很容易实现上述功能。

- PixelsPerInch：设置屏幕分辨率。
- Scaled：是否按照屏幕分辨率调整窗体。

如果将 TForm 组件的 Scaled 属性设为 True，则应用程序在启动时会自动根据运行环境下的 PixelsPerInch 值调整窗体和组件的大小。

2.4　窗体的操作方法

2.4.1　窗体对象的创建与释放

- Create

原型：constructor Create(AOwner:TCcomponent);override;

该方法是一个构造函数，用来创建类的实例（对象）。Create 方法是将对象产生在内存中，并没有显示在窗体上，如果要显示在窗体上，要使用 Show 方法。示例如下：

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
  Form2: TForm;
begin
  Form2 := TForm.Create(nil);
  Form2.Show;
end;
```

- Free

原型：procedure Free;

该方法用于销毁动态产生的 Tform 对象，并且释放其相关内存。

- Destroy

该方法是窗体的析构函数，用来释放窗体资源。在使用“Destroy”方法释放对象时，如果对象为空，会引发异常，所以在程序中通常使用“Free”方法释放对象所占的资源，“Free”方法会检查对象是否为空，如果不为空，调用“Destroy”方法，然后进行内存释放的工作。实际编程中很少直接用 Destroy 方法释放窗体，而是使用 Free 进行窗体的释放。

- Close

执行该方法将关闭窗体。当此方法被调用时，系统会触发 OnCloseQuery 事件，并通过 CanClose 判断窗体是否可以关闭。

- CloseQuery

该方法用于关闭窗体，但在关闭窗体前先询问是否可以关闭窗体。

2.4.2 窗体对象的显示与隐藏

- Show

该方法用于显示窗体（对象）。Show 方法将对象的 Visual 属性设为 True，同时将对象显示在屏幕最上层。示例如下：

```
  Form2 := TForm.Create(nil);
  Form2.Show;
```

如果要显示尚未产生的窗体，则会发生错误。

- ShowModal

该方法与 Show 方法类似，用于显示窗体（对象）。但使用 ShowModal 方法显示的窗体，系统将停留在这个窗口，直到关闭该窗体，系统才继续执行。

- Hide

该方法是用于隐藏窗体。

2.4.3 窗体对象的焦点设置

- SetFocus

该方法可以让窗口控件获得焦点，如果控件不可见或不可用，调用该方法会触发异常。

- Focused

该方法判断对象是否获得了焦点。示例如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if  Not Edit2.Focused  then
```

```
    Edit2.SetFocus;
end;
```

2.4.4 对象控制的操作方法

- RemoveComponent

该方法从窗体对象的组件列表中删除指定的组件，也就是解除窗体对象与组件的宿主关系。示例如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit5 := TEdit.Create(Panel1); //建立Edit5与Panel1的宿主关系，即Panel1是Edit5的所有者，
  //负责Edit5资源的释放
  Edit5.Parent := self;  // 建立窗体对象与Edit5的父子关系，即窗体对象是Edit5的父亲，负责
  //Edit5的显示
  Edit5.Show;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  Panel1.RemoveComponent(Edit5);//解除Edit5与Panel1的属主关系
  Panel1.Free; //Panel1释放后，Edit5并没有被释放
end;
```

- RemoveControl

该方法从窗体对象的控件列表中删除指定的控件，换言之，就是解除窗体对象与控件的父子关系。示例如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit5 := TEdit.Create(Self);
  //建立Panel1与Edit5的父子关系， Panel1是Edit5的父亲，负责Edit5的显示
  Edit5.Parent := Panel1;
  Edit5.Show;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  Panel1.RemoveControl(Edit5); //解除Edit5与Panel1的父子关系，Edit5不可见，也没有被释放
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
  Edit5.Parent := Self; //建立窗体对象与Edit5的父子关系，将Edit5显示在窗体中
end;
```

2.5 窗体的事件处理

2.5.1 窗体激活时产生的相关事件

窗体激活时，将产生一系列事件。这些事件都是按一定顺序产生的，下面按事件产生

顺序介绍这些事件。

- OnCreate：在窗体创建时发生该事件。一般用于初始化组件和窗体成员变量等。
- OnShow：当窗体显示时触发该事件。一般处理组件或窗体的显示变化。
- OnActivate：当窗体激活（获得焦点）时触发该事件。一般处理窗体获得焦点时的显示变化。
- OnResize：当窗体大小变化时触发该事件。一般用于处理用户调整窗体大小时的组件变化。
- OnPaint：当窗体画面重新绘制时触发该事件。

OnCreate、OnShow 和 OnActivate 这几个事件很容易混淆，一定要注意它们的产生顺序。另外，OnCreate、OnShow 只会触发一次，而 OnActivate 会在每次激活时（即焦点从其他窗体或程序跳转到本窗体时）产生。

在应用程序中，如果窗体作为自动创建窗体，那么窗体在程序启动时就已创建了，创建代码是 Delphi 自动产生的。打开项目管理器，就可以看到自动创建的代码程序。如“工资管理系统”中，选择“View”/“Units”菜单，在弹出对话框中选择“gzgl”文件，将打开项目管理文件。可以看到程序自动创建了窗体 Form1 和数据模块窗体 Data3，代码如下：

```
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TData3, Data3);
  Application.Run;
end.
```

对于自动创建窗体，调用时只需使用 Show 方法将其实现即可。自动创建窗体可以通过“Project”/“Options”菜单调出 Options 选项卡的 Forms 页进行设定。

如果窗体不是自动创建窗体，那么调用窗体时需要创建该窗体。如在“工资管理系统”中选择“员工浮动工资管理”菜单，将弹出“员工浮动工资管理”窗口。创建代码如下：

```
procedure TForm1.N6Click(Sender: TObject);
begin
  Application.CreateForm(TForm2,Form2);
  Form2.ShowModal;
  Form2.Free;
end;
```

图 2.9 是创建和显示窗体与 OnCreate、OnShow 和 OnActivate 的关系。

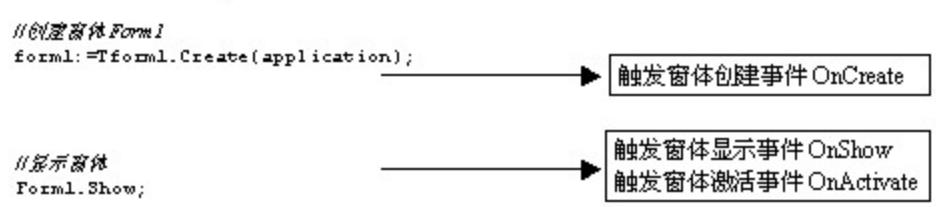


图 2.9　OnCreate、OnShow 和 OnActivate 的关系

2.5.2　窗体关闭时产生的相关事件

窗体关闭时，将产生一系列事件。这些事件都是按一定顺序产生的，下面按事件产生

顺序介绍这些事件。

- OnCloseQuery：在窗体关闭之前触发该事件，提示是否关闭窗体。
- OnClose：在窗体关闭时触发的事件，选择关闭方式。
- OnDeactivate：在窗体失去焦点时触发该事件。处理焦点失去时的显示变化。
- OnDestroy：处理窗体释放时触发的事件，处理内存清理工作。

当调用窗体的 Close 方法时触发 OnCloseQuery 事件。OnClose 方法是使用系统按钮关闭窗体时触发。利用 OnCloseQuery 事件可以在关闭窗体前提示用户保存数据。如在商品入库时，如果用户录入数据时不小心选择了“关闭”按钮，程序弹出对话窗口，提示用户入库数据没有保存，是否确认退出。代码如下：

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  if messagedlg('入库数据还没有保存，确认要退出吗？',mtconfirmation,[mbyes,mbno],0)=mryes then
                Canclose:=true else Canclose:=false;
end;
```

OnClose 事件是当窗体被关闭时触发的。主要用于决定用户关闭窗体的方式。代码声明如下：

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
```

其中参数 Action 用来决定是否真的关闭窗体，Action 可设置的参数如下。

- caNone：窗体不允许被关闭。
- caHide：窗体隐藏起来，不关闭。相当于调用窗体的 Hide 方法。
- caFree：关闭窗体并释放内存空间。相当于调用窗体的 Free 方法。
- caMinimze：窗体最小化。

OnDeactivate 在窗体失去焦点时触发该事件，发生在窗体关闭时或用户将焦点交给其他程序或窗体时。利用该事件可以设计在窗体焦点失去的一些控制。例如设计英语单词朗读的窗体，当焦点在该窗体上时，能按设定不断朗读选定的英语单词，但当焦点跳转到其他程序时，我们希望停止英语单词的朗读。可以在 OnDeactivate 事件中进行控制。

2.5.3 鼠标相关事件处理

在窗口应用程序中，鼠标和键盘的作用相当重要，对计算机的大部分操作都需通过鼠标和键盘来完成。而鼠标事件是应用程序中最重要的事件之一。窗体中与鼠标相关的事件如下。

- OnClick：用户在窗体中单击鼠标时会触发该事件。
- OnDblClick：用户在窗体中双击鼠标时会触发该事件。
- OnMouseDown：处理鼠标按下时引起的事件。
- OnMouseMove：处理鼠标在窗体上移动时触发的事件。
- OnMouseUp：处理释放鼠标按钮时触发的事件。
- OnMouseWheel：处理鼠标轮滚动时的事件。
- OnMouseWheelDown：处理鼠标轮向下滚动时的事件。
- OnMouseWheelUp：处理鼠标轮向上滚动时的事件。

当用户在窗体上单击鼠标时，按键会触发 OnClick 事件和 OnMouseDown 事件。释放按键后会触发 OnMouseUp 事件。如果按住鼠标在窗体上移动，会触发 OnMouseDown、OnMouseMove 和 OnMouseUp 事件。利用这几个事件可以很容易地获得鼠标的位置和状态。

其中，OnClick 事件只与鼠标左键有关，OnMouseDown、OnMouseMove 和 OnMouseUp 事件不但可以与鼠标左键有关，还可以与鼠标右键、中间键或键盘中 Alt、Ctrl、Shift 键有关。OnDblClick 事件与 OnClick 事件类似，只不过对应两次连续的鼠标左键单击。这两次单击的时间间隔较短，系统将把这样的双击动作当作一个事件来处理。鼠标事件处理过程的形式如图 2.10 所示。

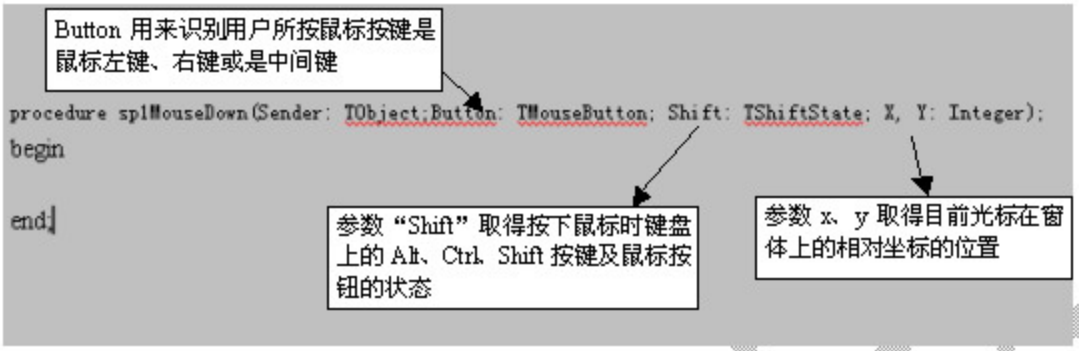


图 2.10　鼠标事件处理过程的形式

现在市场上的鼠标一般都带有滚轮，用户可以通过滚轮方便地浏览信息。Delphi 提供了 OnMouseWheel、OnMouseWheelDown 和 OnMouseWheelUp 共 3 个处理鼠标轮滚动时的事件。

这 3 个鼠标滚轮事件与普通鼠标键消息类似，但它们主要用于滚动操作，因而它们会自动转换为滚动事件（一般应用于窗体或组件的滚动条）。对于系统来说，鼠标键的意义是可以交换的：鼠标的左键和右键可以交换，一次鼠标中键的单击可以使用鼠标左右键的同时单击模拟等等。重要的不是鼠标的实际键在鼠标位置的左边还是右边，而是系统解释它的方式。图 2.11 是固定资产主程序窗体中鼠标事件的应用举例。

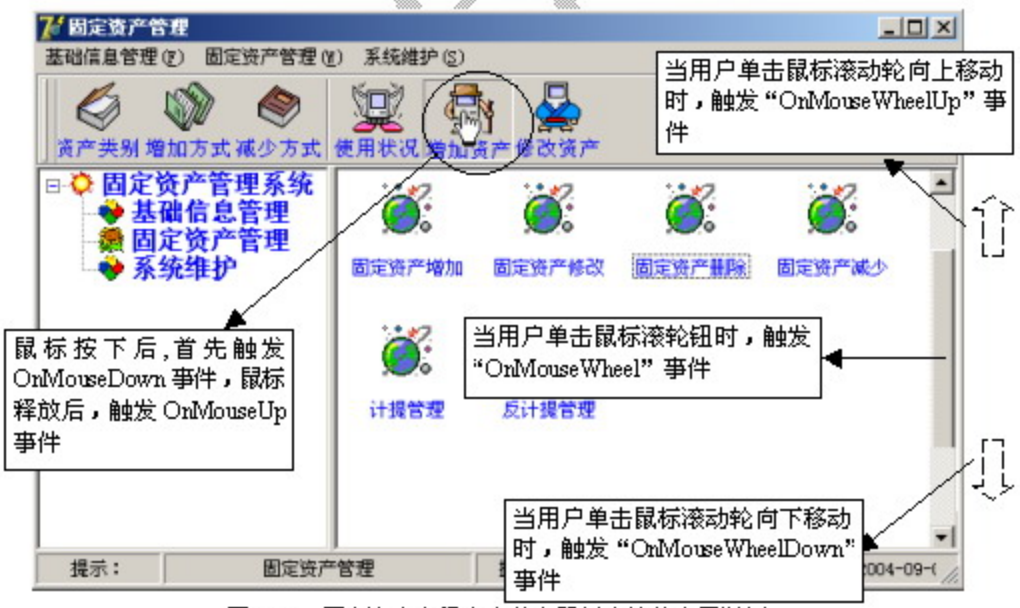


图 2.11　固定资产主程序窗体中鼠标事件的应用举例

2.5.4　键盘相关事件处理

Delphi 提供了详尽的键盘事件。但一般很少用到窗体上。键盘事件大多数操作都是由特定

组件（例如 TEdit、TMemo 等）完成的，用户的应用程序应该包含了这些输入控制组件。此外，如果用户输入了某个键盘快捷键，系统会自动将它们全部交给应用程序的菜单去处理了。

当然，响应窗体上的键盘事件也并不是完全没有，例如在一些程序中，如果用户计算机没有鼠标或暂时鼠标不能用，就可以为窗体程序设计输入某个组件键关闭窗体或程序。TForm 组件的键盘事件如下。

- OnShortCut 事件：在用户单击了快捷键时触发此事件。
- OnKeyDown 事件：在用户按下某键时触发此事件。
- OnKeyPress 事件：在用户单击某键时触发此事件。
- OnKeyString 事件：在用户输入某个多字节字符时触发此事件。
- OnKeyUp 事件：在用户释放某键时触发此事件。

因为键盘键上按键较多，功能组合较复杂，捕获键盘事件要比捕获鼠标事件困难。键盘事件的最常用事件为 OnKeyDown 和 OnKeyPress。下面首先介绍 OnKeyDown 事件。

OnKeyDown 事件的原型为：

```
procedure FormKeyDown(Sender:TObject;var Key:Word;Shift:TShiftState);
```

键盘事件中最重要的参数为“Key”和“Shift”。“Key”参数在 OnKeyDown 事件中的类型为 Word 类型，返回键盘按下的键盘扫描码。“Shift”参数用来判断用户是否按下 Alt、Ctrl、Shift 键。下面代码使用组合键 Alt 和回车键关闭窗体。

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (shift=[ssAlt])and (Key = VK_RETURN) then   close;
end;
```

键盘事件总是在键盘事件被系统内部处理前触发，最先响应的是 OnKeyPress 事件，此事件在用户键入某键时触发，OnKeyPress 事件中最重要的是 Key 参数，用来表示用户刚才输入的字符键。只有 ASCII 字符才可以由此 Key 参数传递，OnKeyPress 事件只能响应 ASCII 字符键。下面代码限制在窗体中使用除退格键 BackSpace、删除键 Delete、小数点和数字键之外的字符。

```
procedure Tform1.FormKeyPress(Sender: TObject; var Key: Char);
var
  mm: Boolean;
begin
  mm := (Key<#8)or(Key>#8)and(Key<#46)or(Key>#46)and(Key<#48)or(Key>#57);
  if mm then
    Key := #0;
end;
```

2.5.5 拖动相关事件处理

按下鼠标键不放，移动鼠标到指定位置，然后释放鼠标键，这个过程称为鼠标的拖动。在 Windows 应用程序中，鼠标拖动操作是很常用的操作。Delphi 为开发人员提供了用于拖动过程的可视化组件，只要编写两三行代码就能实现拖动的功能。另外，Delphi 组件也提供了 DragMode 属性和 OnDragDrop、OnDragOver 事件来实现此功能。

- OnDragDrop

处理用户释放在窗体中被拖动的对象时触发的事件。

- OnDragOver

处理在窗体上拖动一个控件触发的事件。

- OnDockDrop

当其他的对象移入到窗体对象时触发该事件。该事件只有在“DockSite”属性为“True”时触发。

- OnDockOver

当其他的对象在窗体对象上拖动时触发该事件。该事件只有在“DockSite”属性为“True”时触发。

2.6 应用程序窗体组成

一个应用程序中使用的窗口通常由启动提示窗口（Splash Form）、程序主窗口（Main Form）、对话信息窗口（Dialog/Message Form）以及应用程序窗口等组成。对于各种窗口在执行程序时被加载内存的应用顺序如图 2.12 所示。

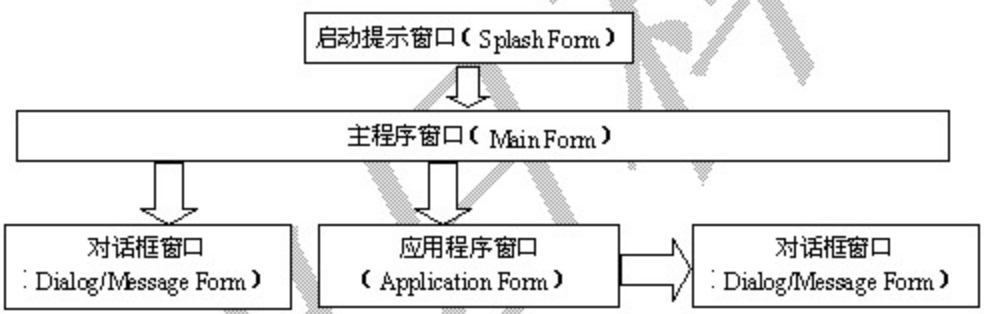


图 2.12 各种窗口被加载内存的应用顺序

启动提示窗口是程序执行时最先被加载和显示的窗口。一般使用程序代码声明建立，在程序主窗口完全加载内存后便自动删除内存；主程序窗口是应用程序的控制中心，通过主程序窗口的菜单、工具栏按钮可以调用相关应用程序窗口；对话信息窗口是应用程序和用户通信的窗口，用户可以通过对话信息窗口输入信息、阅读提示、选择选项等；应用程序窗口是基础应用程序的重要组成部分，应用程序中所有任务功能都要通过应用程序窗口完成。应用程序窗口分为自动加载内存与利用程序代码声明建立两种。可以通过设置 Form 选项卡进行设置。但是不论哪一种都是在程序主窗口加载内存后，才会被加载内存。

1. 启动提示窗口（Splash Form）

启动提示窗口（Splash Form）也称欢迎窗口。用来显示应用程序的版权、公司和作者等信息。尤其当开发的应用程序较大，启动时需要主动或被动加载大量 Dll 文件，需要花费较长时间才能显示时，为了缓解程序过长的启动等待，可以添加启动提示窗口，提示用户正在加载程序文件。

启动提示窗口需要使用程序代码声明建立程序。建立步骤如下。

（1）在 Delphi 环境下打开“客户管理系统”，选择“File”/“New”/“Form”菜单，新建空白窗体文件，将新窗体 Name 属性设为 Splashform，保存单元文件为 Splashunit.pas。

（2）在新建窗体中加入 Image 组件和 Label 组件，新建窗体、Image 组件和 Label 组件属性设置如图 2.13 所示。

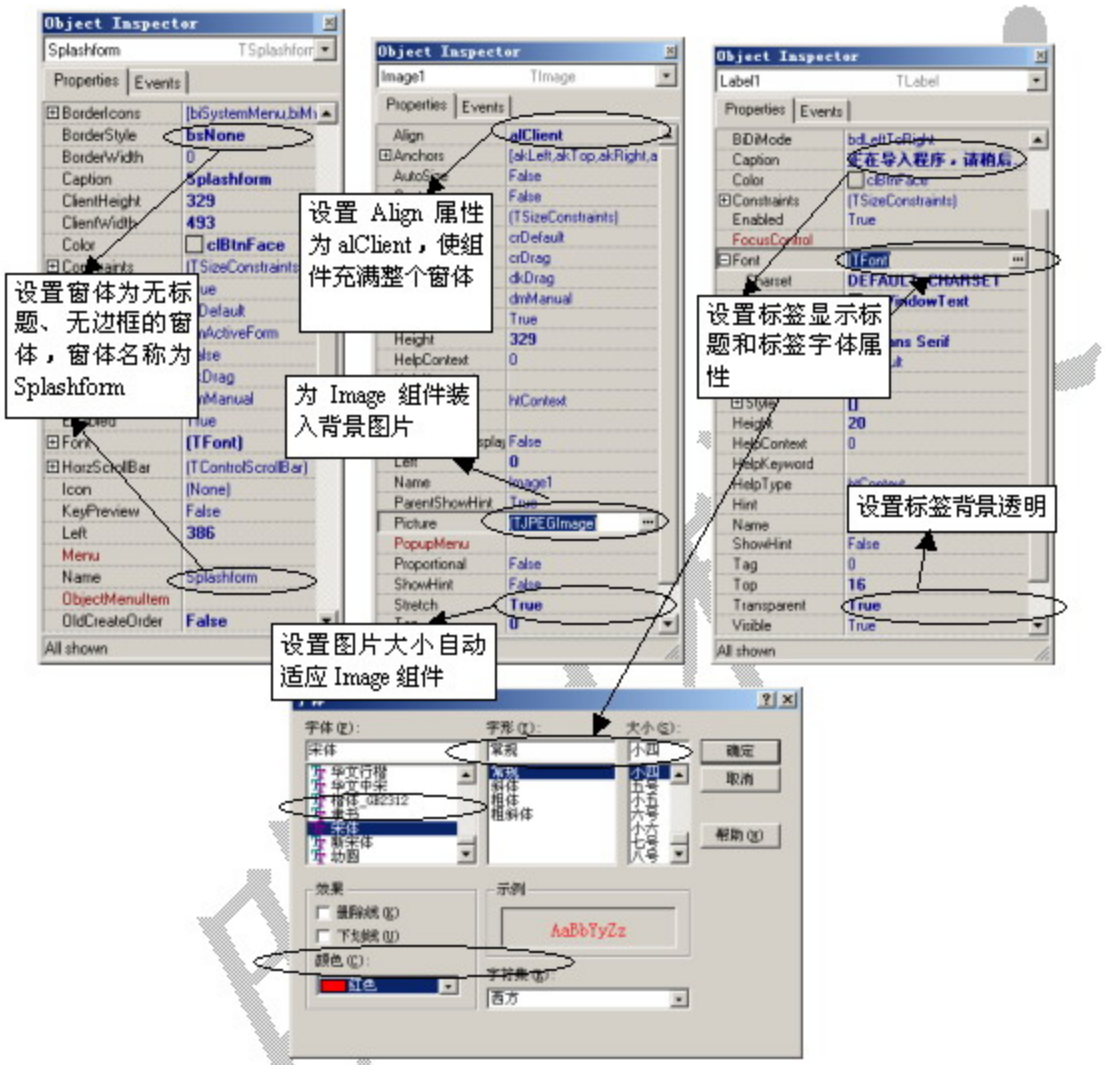


图 2.13　属性设置

（3）打开“Project”/“Options”选项的“Forms”选项卡，将 Splashform 移至 Available forms 中（因为要使用代码声明建立启动提示窗口，所以应设置该窗体为非自动创建窗体），如图 2.14 所示。

（4）选择“Projects”/“View Source”菜单项，打开项目文件，在项目文件的程序代码区加入如下代码，代码加入位置如图 2.15 所示。

```
SPLASHFORM:=TSPLASHFORM.Create(Application);
SPLASHFORM.SHOW;
SPLASHFORM.update;
SPLASHFORM.hide;
SPLASHFORM.free;
```

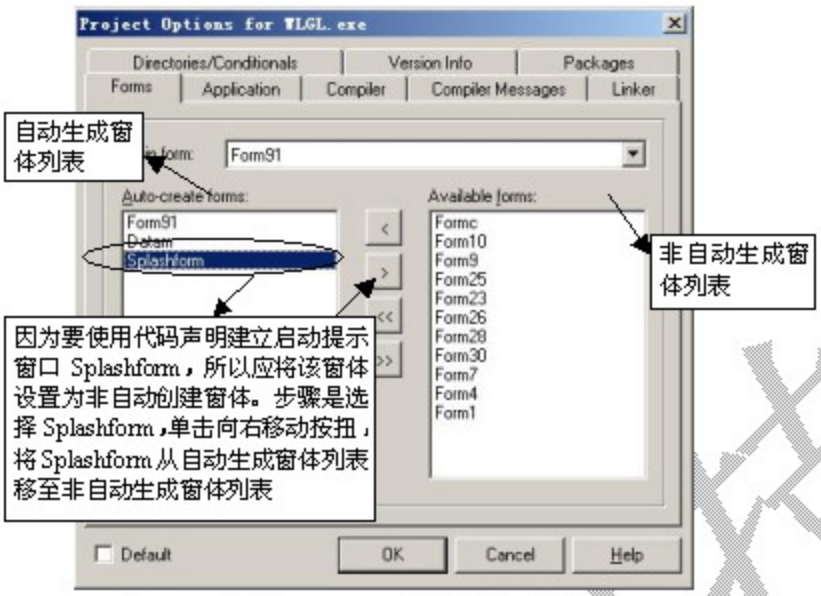


图 2.14　设置 Splashform 窗体为非自动创建窗体

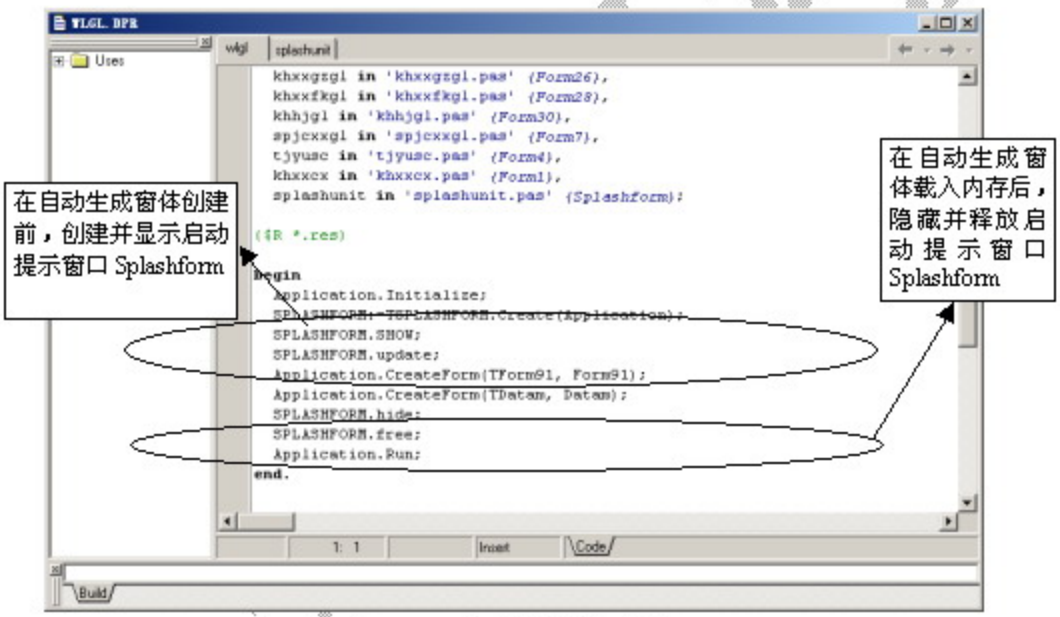


图 2.15　为工程文件加入代码

（5）运行程序，启动提示窗口 Splashform，运行效果如图 2.16 所示。



图 2.16　启动窗体运行效果

2. 程序主窗口（Main Form）

程序主窗口即应用程序的主界面窗口，主要根据操作员权限设置系统菜单、工具栏、快速导航条的相应调用权限，并记录操作员工作日志，进行程序状态提示等功能。主程序窗口是应用程序的控制

中心，通过主程序窗口可以打开相应的功能窗口，从而完成相应的功能。主程序窗口一般包含以下功能。

- 通过菜单、工具栏、快速导航栏等调用相应子程序。
- 显示当前系统时间、当前操作员和系统帮助信息。
- 根据操作员权限设置相应菜单、工具按钮及导航按钮是否有效。
- 提供网络在线帮助等。

主界面设计的好坏往往影响到软件的整体形象，因此必须首先处理好主界面的设计问题。同时，主程序也是程序控制的核心，操作权限、程序功能调用等都要通过主程序调用实现。下面是“医药管理系统”程序的主界面窗口（如图 2.17 所示），其大致可以分为 4 个功能区域。



图 2.17 医药管理系统程序的主界面窗口

打开 Delphi 时，会自动创建一个工程，并生成一个窗体，默认情况下该窗体即是应用程序的主窗体。在 Delphi 中，应用程序可以有多个窗体，但只能有一个主窗体，主窗体是应用程序第一个创建的窗体，如果主窗体关闭，应用程序便会结束。用户可以在程序中随意将一个窗体设置为主窗体，方法是单击“Project”/“Options”菜单项，打开“工程选项”窗口，打开“Forms”选项卡（如图 2.18 所示），设置步骤如下。

（1）单击 Main form 组合框右侧的下拉按钮，将弹出窗体列表。

（2）在窗体列表中选择要设置成主窗体的窗体。

（3）单击 OK 按钮完成设置。

3. 对话信息窗口（Dialog/Message）

对话框信息窗口是应用程序和用户通信的窗口，其作用是用户在此可以输入信息、阅

读提示、选择选项等。对话信息窗口包括对话窗口和信息窗口。对话窗口通常为了某个单一目的让用户输入选择并将数据返回，而信息窗口通常是用来显示程序执行信息。对话信息窗口典型形式如图 2.19 所示。

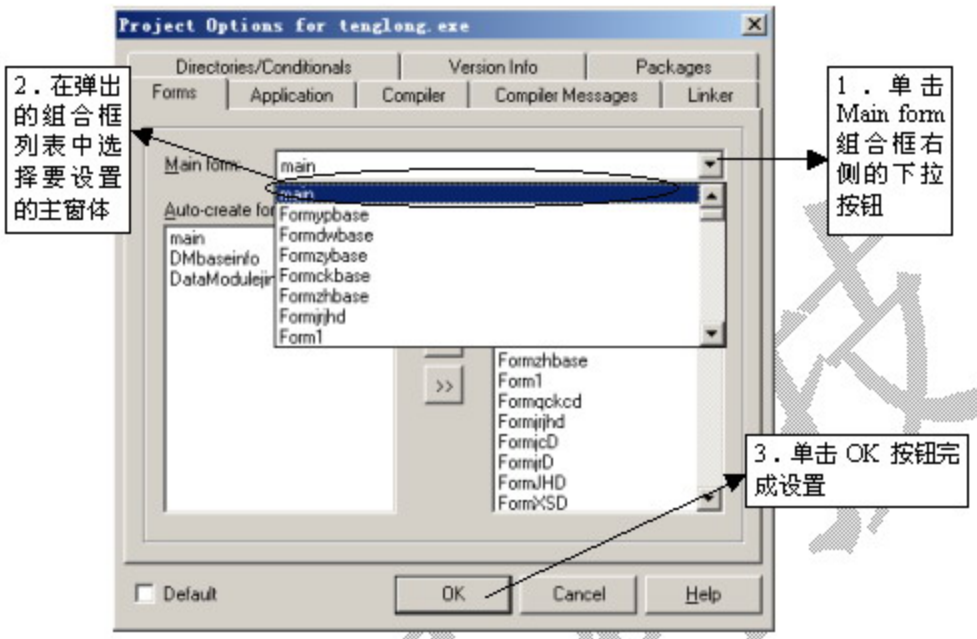


图 2.18　工程选项窗口

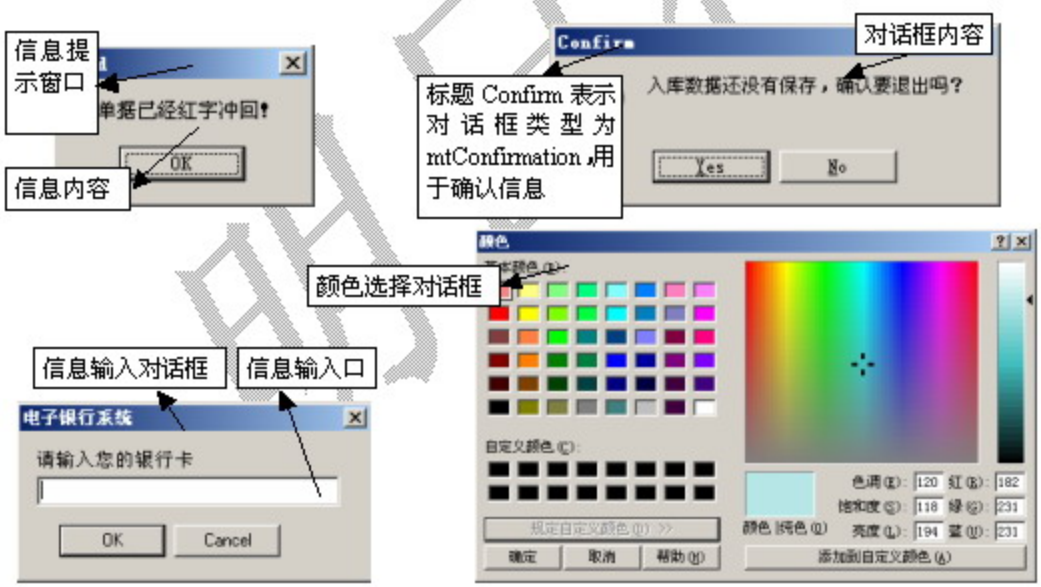


图 2.19　对话信息窗口典型形式

Delphi 提供使用函数调用方式的对话信息窗口，也可以使用组件工具栏 Dialogs 选项卡中的对话框组件。Delphi 内建的对话信息窗口包含下列几种。

- ShowMessage 函数
- MessageBox 函数
- MessageDlg 函数与 MessageDlgPos 函数

- InputBox 函数
- InputQuery 函数
- LoginDialog 函数与 LoginDialogEx 函数
- SelectDirectory 函数

以上都是使用函数调用方式产生的对话信息窗口，Delphi 还提供另外一种使用组件工具栏上内建 Dialogs 选项卡中的组件产生的对话窗口。

基本上 Dialogs 选项卡中的组件都是不可视组件，执行时期必须通过对象方法的调用才能显示对话窗口，设计时期则在对象图标上双击可以先看到对话窗口的模样。

4. 应用程序窗口

应用程序窗口是程序的主要部分，它用于实现应用程序要实现的各个功能。一个应用程序，一般包含若干需要解决的任务，要用一个窗体完成所有任务是不可能的，通常需要分解的任务按照功能结构和任务的复杂程度，使用一个或多个与之关联的窗体来完成。这些应用程序窗口主要通过主程序窗口进行切换。例如在“配送管理系统”中，需要完成的工作任务有商品入库管理、配送申请管理、配送审核管理、配送开票管理、配送查询管理、商品信息维护管理、客户信息维护管理及操作员维护管理等。其中，商品入库管理、配送申请管理、配送审核管理、配送开票管理可以分别使用一个窗体完成相应功能，而客户信息维护管理因为涉及客户信息的录入、修改、查询等功能，如果放在一个窗体内实现起来比较麻烦，也不利于用户使用，因此可以设置一个客户维护管理主窗体，用于显示和调用功能。而添加、修改及查询功能可以通过新建窗体来完成。客户维护管理各窗体关系如图 2.20 所示。

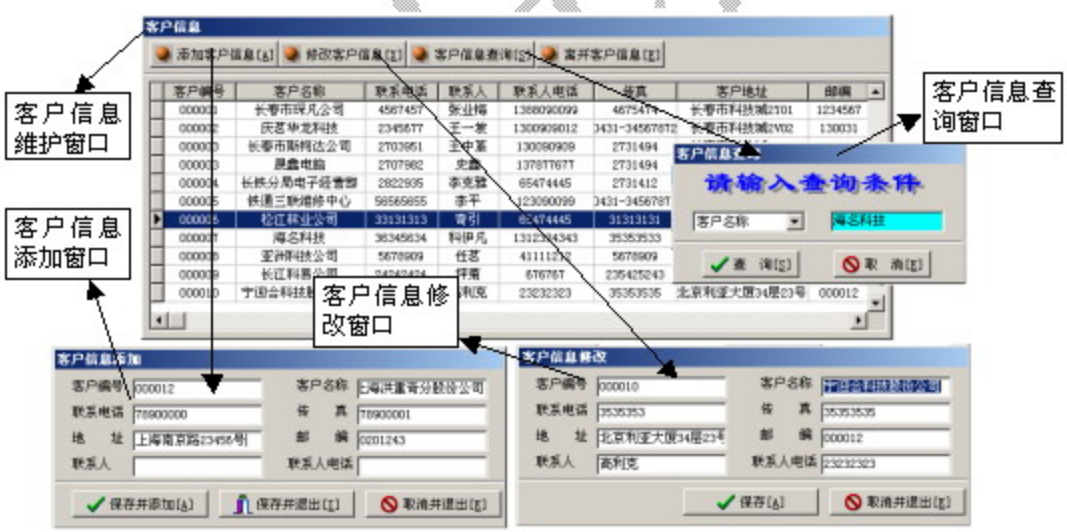


图 2.20　客户维护管理各窗口显示

应用程序窗口通常设有工具栏、菜单栏和状态栏，它们都能执行应用程序的某部分功能。一般情况下，打开应用程序后，屏幕上某个时刻只显示一个窗体（主窗体除外），其他窗体要么隐藏，要么从内存中清除。为了提高程序运行速度，不提倡使用隐藏窗体的方法。因为虽然窗体隐藏后看不见窗体了，但它仍存储在内存中，当打开窗体过多时，可能造成内存紧张。应将窗体从内存中清除后，调用时，只需再调入该窗体。当某一功能窗口成为焦点窗口

后，其他窗口一般处于锁定状态，当该窗体失去焦点时，其他功能窗体才能获得焦点。

2.7 应用软件开发流程

一个应用软件开发项目，从最初的需求分析到最后的系统维护一般要经过以下 5 个阶段。如图 2.21 所示。

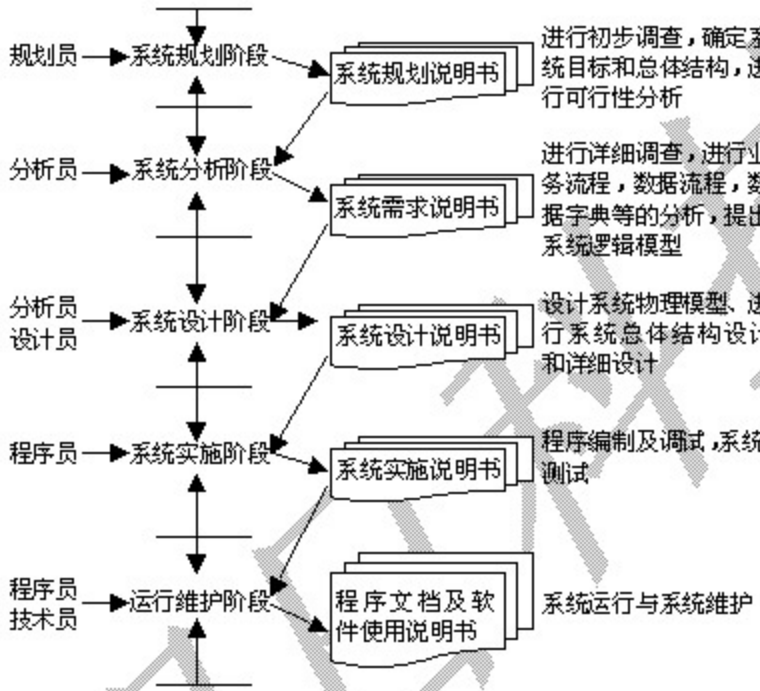


图 2.21 软件开发流程

首先是系统规划阶段。在规划阶段首先要进行项目开发的初步调查，作为跟客户谈判的项目人员，要初步了解项目的范围和要求，系统的目标及规模、管理体制和管理水平；要花多少钱、多少人力、用多少时间。另外，还必须进行系统可行性分析。系统软件开发是一项耗费较大人力、物力、财力的工程，在系统开始投入设计之前，必须认真地进行可行性分析，从经济、技术等方面论证其建立的必要性和可能性。要根据客户单位的具体情况确定系统的目标、规模以及系统建立的方式，以尽量少的消耗最大限度地满足客户单位的需要。反之，如果不进行认真周密的可行性分析，盲目进行开发，将来开发完成的软件有可能因外部环境的限制而无法使用，或出现软件的维护费大于收益的现象。

在对系统进行初步调查和可行性分析之后，就要进入系统分析阶段。首先要弄清楚这个系统到底是干什么的，为客户到底解决什么问题，怎么样用这个软件。然后根据这些分析形成面向软件开发人员的包括数据流图、数据字典、输入输出项目和要求等内容设计软件逻辑模型，编制软件需求说明书，制定系统设计的初步计划。类似于建设高楼，大楼是做什么用的，设计成什么样，多少层，每层里面多少间屋子，屋子里面有什么设备，这些都是盖楼前需要分析和确定的。这时，不必考虑使用什么材料，哪个工程队进行建设等具体的开发问题。

系统分析之后，要进行系统设计。如果系统分析阶段的任务是解决“干什么”的问题，那么系统设计阶段的任务是确定“怎么干”。系统设计工作由总体设计和详细设计两部分组成。总体设计是软件开发人员根据软件需求说明书的要求，运用结构化程序设计思想，将软件自上而下逐层分解成多个软件模块，直到分解成每一个模块只具有单一的功能，能用一个或几个程序实现的树状结构为止。总体设计阶段还要定义各模块的数据传递关系，设计软件的编码方案、文件存储策略、输入输出格式，以及硬件和系统软件配置，最后编制概要设计说明书。详细设计就是对总体设计中划分的每个模块再进行详细定义和说明。它包括定义每一模块的详细功能、输入数据、使用文件及使用方式，确定输出内容及格式，模块实现的详细算法，每一模块的程序构成等。详细设计是软件功能、结构实现方法的最详细说明，是程序设计的依据。详细设计的最终成果是编制详细设计说明书。也就是说这个项目管理产品该做什么样的东西，有什么样的功能，但是具体下面怎么样去实现，比如说构架怎么回事，它用什么样的语言，这个不用管。就像建设高速公路，作为负责人，只需进行高速公路的总体设计，也就是高速公路的长度、宽度、公路等级等的设计。至于修建高速公路时某个路段需要石头多少，水泥多少，多少人施工，这个不需要负责人管，这是真正建设公路的人需要知道的。

接下来就要进入系统实施阶段，开始编写程序代码了。到了这个时候程序才真正开发了。软件开发人员根据详细设计说明书自上而下地将每一模块用指定的程序设计语言编写成源程序代码，并进行调试，保证运行的正确性，同时编制程序说明书。程序设计由程序编制和程序调试两个阶段组成。任何软件开发人员都不可能保证编制的程序一次运行正确，都要经过反复调试。程序调试的过程就是发现错误并改正错误的过程，直到软件开发人员确定全部程序无任何语法错误或逻辑错误，并满足系统设计的各项要求为止。对于一般应用程序，程序调试的工作量等同甚至超过编制过程的工作量，而且程序调试是一项非常艰苦的工作。在制定开发计划的时候也必须同时制定相应的测试计划。

软件通过测试后交付用户使用，软件开始进入运行维护阶段了，软件投入使用后，可能会出现很多问题，软件维护人员要对软件进行不断的修改、补充和日常维护，使软件运行稳定并不断完善。

用户图形界面设计

第3章

3.1 可视化用户界面设计

随着 Windows 操作系统在 PC（Personal Computer）中的流行，窗口程序设计在编程领域异军突起，它以其简单、方便的操作备受用户的青睐，但这对当时的程序开发人员来说也许是一个灾难，因为当时并没有 Windows 框架（例如 MFC、OWL、VCL），开发人员要创建一个简单的窗口就需要使用很多 API 函数，而且在窗口函数中需要处理大量的 Windows 消息，要设计一个界面美观的窗口更是难上加难。后来，随着 Borland、Microsoft 等一些厂商推出相应的 Windows 框架后，Windows 程序开发才变得简单、快捷，在这些框架中提供了丰富的组件，开发人员可以直接使用这些组件设计应用程序，而无须再处理大量的窗口消息，这大大简化了 Windows 程序设计的难度，同时提高了程序的开发效率，尤其是一些 RAD 开发工具的出现，程序员只需在窗体中放置一些组件，然后调整组件的外观属性便可以设计出美观的界面，甚至不需要写任何代码。

3.1.1 界面设计基础

界面设计一般是以选定的界面支持系统为基础，利用它所支持的界面成分，构造一个满意的程序界面。目前，在程序界面的开发中最受欢迎的支持系统是将视窗系统、图形用户接口、编程语言、可视化开发工具与类库融为一体的可视化编程环境。在可视化开发环境中，程序员可以直接使用开发环境提供的各种控件拼凑、绘制所需的界面。在一些 RAD 开发环境（例如 Delphi、Visual Basic）中，程序员甚至不需运行程序，在设计时便能观看到界面的效果。

3.1.2 人机交互界面设计

人机交互界面被定义为一种采用 4 种基本组件（窗口、图标、菜单和鼠标指针）的人机交互形式。最主要的特征是直观操作、提供鼠标操作。

1. 界面要素

- 窗口

窗口是显示设备（例如显示器）中的一个区域，可以打开、关闭、移动或改变大小。

- 菜单

菜单用于显示一组操作或命令，供用户选择。每一个操作或命令对应一个菜单项，菜单项可以包含文字或图标。通常情况下，菜单是通过窗口来显示的。

- 图标

图标是一个显示设备的区域，以图形化的方式表示对象。

- 指针

指针是一个图形，用以对指定设备输入到系统的位置进行可视化描述。

- 其他

包括按钮、面板、滚动条等元素。

2. 设计准则

- 以用户为中心

开发人员在开发程序时，应以用户需求为核心，因为再完美的应用程序如果满足不了用户的需求，它也只能是一种修饰品，没有任何的实际用处和商业价值。

- 使用方便

用户在完成一次与程序交互时，要保证操作尽可能少，将键盘和鼠标的点击次数减到最少。

- 减少人脑记忆负担

程序操作要尽量简单，使用户不必记忆大量的操作规则和对话框信息。

- 减少重复录入

对于用户录入过的信息，一段时间后，用户需要重新使用这些信息，能够通过简单操作复用以往的信息，而不需要用户重新录入。

- 一致性

程序的各个部分应在交互方式、操作步骤等方面尽可能保持一致。使用户能够举一反三、触类旁通地掌握对程序的操作。

3.1.3 人机交互界面举例

- 传统的应用程序主界面（如图 3.1 所示）
- 简洁的应用程序界面（如图 3.2 所示）

明日科技



图 3.1　仓库管理系统界面



图 3.2　人事管理系统界面

- 典型的应用程序界面（如图 3.3 所示）



图 3.3　配送管理系统界面

❑ 特殊风格的程序界面（如图 3.4 所示）



图 3.4 流通企业程序界面

3.2 控件概述

3.2.1 控件分类

初学 Delphi 的人可能分不清控件与组件的区别，在此作者做一下介绍。控件是从组件继承来的，即所有的控件都是组件，反之，所有的组件不都是控件。控件与组件的区别在于控件提供了运行时可见的功能，因此，在控件类中提供了与位置有关的属性、与显示有关的方法以及与鼠标有关的事件等。

在 Delphi 中，控件分为窗口控件与图形控件两大类。窗口控件具有 Windows 句柄，能够获得焦点。图形控件又称为非窗口控件，没有 Windows 句柄，占用的资源要比窗口控件少，此外，图形控件的重画速度要比窗口控件快得多。

3.2.2 控件命名约定

在开发程序时，会使用许多控件，建议更改默认的控件名称，将其更名为具有一定含义的统一的控件名称。这样，不但有利于编写代码，而且会提高程序的可读性。例如图 3.5 所示的窗体界面，可以为编辑框控件分别命名为“OperatorBH”、“OperatorMC”、“OperatorMM”和“OperatorQX”。



图 3.5 操作员信息管理

3.2.3 控件的通用属性

- Align

确定控件在其父类中的对齐方式。

- Color

确定控件的颜色。

- ClientRect

确定控件的客户区域占有的矩形。

- Hint

确定在控件的工具提示中显示的文字。

- Enabled

确定控件是否可用。

- DragKind

该属性指定控件操作时拖动的类型。

- Font

确定控件上文字的字体。

- Parent

确定控件的父类或容器，即控件在哪个窗口控件上显示。

- Text

确定控件上可修改的文字。

- Visible

使控件可见。

3.2.4 控件的通用方法与事件

- Refresh

删除并重画控件。

- Repaint

重画控件。

- Show

使控件可见。

- Hide

使控件不可见。

- Invalidate

迫使重新绘制控件。

- GetControlsAlignment

确定文本在控件内的对齐方式。

- OnCanResize

当调整控件的大小时触发该事件。

- OnClick

当用户单击控件时触发该事件。

- OnContextPopup

当用户右键单击控件或调用弹出式菜单时触发该事件。

明日科技

- OnDblClick

当用户双击控件时触发该事件。

- OnDragDrop

当用户释放被拖动的对象时触发该事件。

- OnDragOver

当用户在控件上拖动对象时触发该事件。

- OnEndDock

当用户结束拖入操作时触发该事件。

- OnEndDrag

当用户结束拖动操作时触发该事件。

- OnMouseDown

当用户在控件上按下鼠标按键时触发该事件。

- OnMouseMove

当用户在控件上移动鼠标指针时触发该事件。

- OnMouseUp

当用户在控件上释放鼠标按键时触发该事件。

- OnMouseWheel

当鼠标滚轮滚动时触发该事件。

- OnMouseWheelDown

当鼠标滚轮向下滚动时触发该事件。

- OnMouseWheelUp

当鼠标滚轮向上滚动时触发该事件。

- OnResize

当控件被调整大小后触发该事件。

- OnStartDock

当用户开始拖动对象时触发该事件。

- OnStartDrag

当用户开始拖动对象时触发该事件。

注意：OnEndDock 事件与 OnStartDock 事件只有在控件的 DragKind 属性为“dkDock”时才可能发生。

3.2.5 关于 Tab 键次序

将窗口控件放置在一个窗体或容器中时，设置 Tab 键顺序可以使每个控件在创建时被有序化。用户在运行程序时，可以按 Tab 键切换窗口控件的焦点。如果想修改窗口控件的 Tab 键顺序，可以设置 TabOrder 属性，Tab 键的控制次序是从 0 开始的，因此，Tab 键控制次序中的第一个控件的 Tab 键顺序为 0。窗口控件有一个 TabStop 属性，该属性确定用户按 Tab 键是否可切换焦点。如果该属性为“False”，窗口控件不处于 Tab 键控制顺序中，TabOrder 属性无意义。

3.2.6 关于焦点

前面介绍过，只有窗口控件才具有焦点。当窗口控件获得焦点时，该控件将成为当前

控件。用户可以通过转移窗口控件的焦点使其他窗口控件成为当前控件。

3.3 使用控件数组

3.3.1 为何使用控件数组

在开发程序的过程中会经常在窗体中放置多个同类控件。例如，在窗体中放置多个编辑框控件。如果要清空这些编辑框中的内容，可能会使用许多语句来为每一个编辑框清空文本，这样做显然很麻烦。如果使用控件数组，将控件数组中的元素指向相应的控件，就可以使用循环语句来清空每一个编辑框中的文本，省去了大量语句。

3.3.2 定义控件数组

定义控件数组与定义普通数组类似，只是数组的类型为控件类型。示例如下：

```
SArrayEdit: Array [0..3]of TEdit; //静态控件数组
DArrayEdit: Array of TEdit; //动态控件数组
```

上面的示例中，分别定义了一个静态控件数组与一个动态控件数组，动态控件数组在使用前需要设置数组的长度，即设置数组的大小。

3.3.3 控件数组应用实例

```
var
  Form1: TForm1;
  SArrayEdit: Array [0..3]of TEdit;
  DArrayEdit: Array of TEdit;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
  i: Integer;
begin
  For i := 0 to 3 do
    SArrayEdit[i].Clear;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  SArrayEdit[0] := Edit1;
  SArrayEdit[1] := Edit2;
  SArrayEdit[2] := Edit3;
  SArrayEdit[3] := Edit4;
  SetLength(DArrayEdit,4);//设置动态数组的长度
end;
procedure TForm1.Button2Click(Sender: TObject);
var
```

```
    i: Integer;
begin
    DArrayEdit[0] := Edit1;
    DArrayEdit[1] := Edit2;
    DArrayEdit[2] := Edit3;
    DArrayEdit[3] := Edit4;
    For i := 0 to 3 do
        DArrayEdit[i].Text := '0';
end;
end.
```

3.4 调整控件布局

3.4.1 统一控件尺寸

1. 按比例调整控件尺寸

在窗体中按住 Shift 键，然后单击窗体中的控件，使控件处于选中状态，进行如图 3.6 所示的操作，按比例调整控件尺寸。

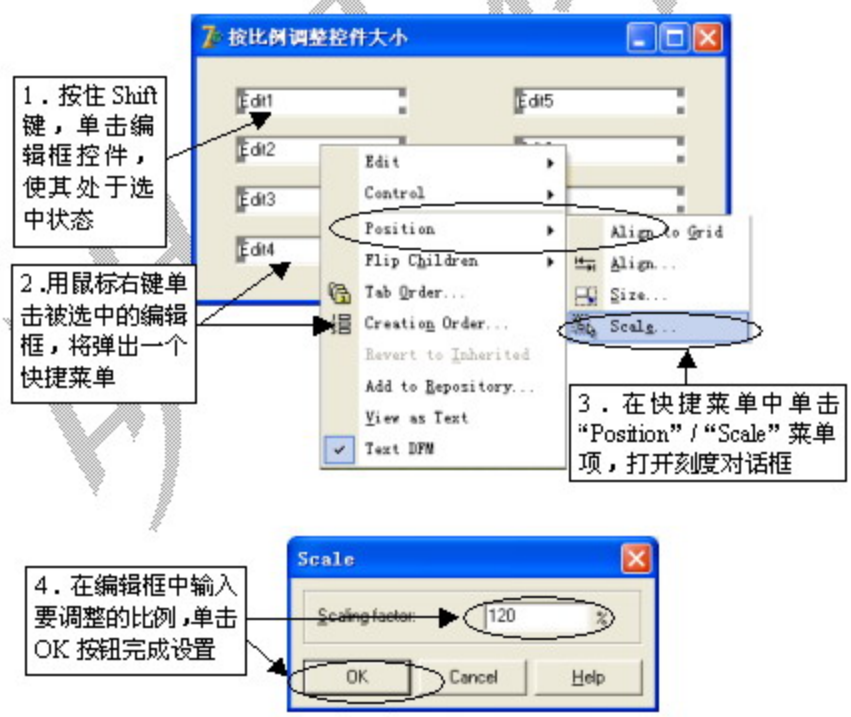


图 3.6 按比例调整控件大小

2. 按尺寸调整控件大小

在窗体中按 Ctrl+A 组合键选中所有控件，进行如图 3.7 所示的操作，按尺寸调整控件大小。

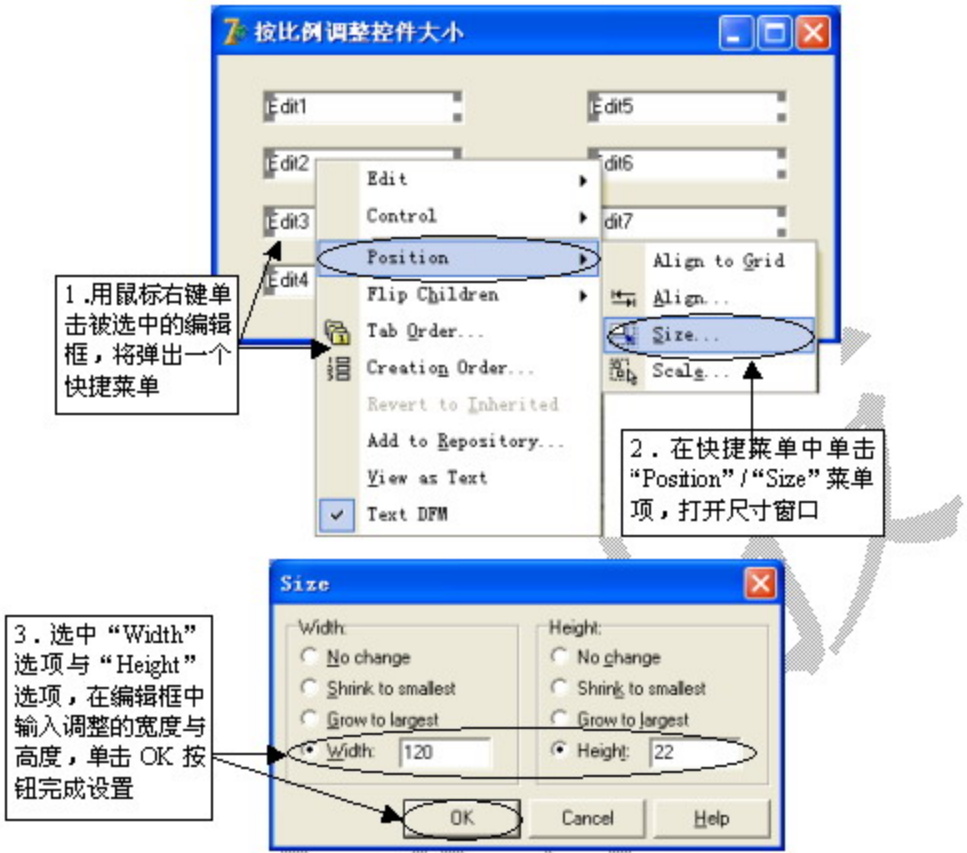


图 3.7　按尺寸调整控件大小

3.4.2　调整水平和垂直间距

1．使用 Alignment 对话框

在窗体中经常会有许多控件，如果用鼠标拖动来调整控件的位置，可能不会很整齐。此时按住 Shift 键，用鼠标单击要对齐的控件，此时，被选中的控件的焦点会变成灰色。然后释放 Shift 键，用鼠标右键单击其中一控件，将弹出一快捷菜单，选中“Position”/“Align”菜单项，将会弹出一对齐面板，在那里可以设置控件的各种对齐方式，如图 3.8 所示。

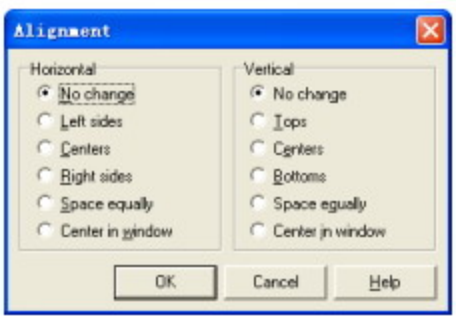


图 3.8　控件对齐方式

控件对齐面板分为两部分，Horizontal、Vertical 即水平对齐设置、垂直对齐设置。在 Horizontal 面板中共有 6 个选项，其意义如下。

- No change

控件的水平位置不改变。

- Left sides

控件居左对齐。

- Centers

控件在水平位置居中对齐，用于调整多个控件的位置。

- Rigth sides

控件居右对齐。

- Space equally

控件水平间距均匀对齐。

- Center in window

控件在水平位置居中对齐。

在 Vertical 面板中共有 6 个选项，其意义如下。

- No change

控件的垂直位置不改变。

- Tops

控件居顶部对齐，用于调整多个组件的位置。

- Centers

控件在垂直位置居中对齐，用于调整多个控件的位置。

- Bottoms

控件居底部对齐，用于调整多个控件的位置。

- Space equally

控件垂直间距均匀对齐。

- Center in window

控件在垂直位置居中对齐。

2. 使用对齐面板

使用对齐面板可以调整选中控件的位置，使其以某个控件为基准对齐。单击“View”/“Alignment Palette”菜单项可以打开对齐面板，如图 3.9 所示。

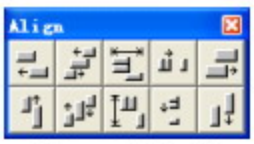


图 3.9　对齐面板

在对齐面板中共有 10 个按钮，含义如下。

- 

被选中的控件以第一个被选中控件的左边线为基准对齐。

- 

水平移动被选中的控件，直到它们的中心位置与第一个被选中控件的中心位置对齐。

- 

移动被选中的控件，使它们对齐到水平线的中心。

- 

水平对齐 3 个或 3 个以上被选中的控件，使中间的控件与外部的控件水平距离相等。

- 

被选中的控件以第一个被选中控件的右边线为基准对齐。

- 

被选中的控件以第一个被选中控件的上边线为基准对齐。

- 

垂直移动被选中的控件，直到它们的中心位置与第一个被选中控件的中心位置对齐。

- 

明日科技

移动被选中的控件，使它们对齐到垂直线的中心。

- 

垂直对齐 3 个或 3 个以上被选中的控件，使中间的控件与外部的控件垂直距离相等。

- 

被选中的控件以第一个被选中控件的底边线为基准对齐。

3.4.3 锁定控件布局

当设计好程序界面时，为防止不小心修改程序界面，可以锁定窗口的布局。单击系统菜单“Edit”/“Lock Controls”菜单项，可以锁定窗口中的控件，这样，就无法拖动窗口中的控件，除非再单击“Edit”/“Lock Controls”菜单项。

3.5 标签控件

3.5.1 主要属性设置

TLabel 组件用于输出一个窗体上的只读文字，常用来标记没有内置标题的控件，其主要属性如下。

- Alignment

该属性确定标题在控件中的对齐方式，共有 3 个属性值，分别为“taLeftJustify”、“taRightJustify”和“taCenter”。

 - taLeftJustify

 表示标题在控件中居左对齐。

 - taRightJustify

 表示标题在控件中居右对齐。

 - taCenter

 表示标题在控件中居中对齐。

- AutoSize

该属性确定标签控件是否根据标题的大小确定其大小。

- Caption

该属性确定标签控件的标题。

- FocusControl

指定一个窗口控件与标签控件关联，当用户按标签控件的加速键时，与之关联的窗口控件会获得焦点。设置标签的加速键可以在标签的“Caption”属性值前输入“&”字符，然后设置“ShowAccelChar”属性为“True”。

- ShowAccelChar

该属性确定标签控件中符号在文本中怎样显示，如果为“True”，在设置加速键时，会在键值下方显示一个下划线，要在文本中显示“&”符号，需要在“Caption”属性中输入两个“&”符号，如果“ShowAccelChar”为“False”，“Caption”属性中的内容会全部作为文本显示，包括“&”符号，并且“FocusControl”属性不可用。

- WordWrap

该属性确定标签控件中的文本是否折行显示。

明日科技

3.5.2 标签控件的实例应用

- 单击 TLabel 控件，为其设置加速键。

```
procedure TForm1.Label1Click(Sender: TObject);
begin
  Label1.Caption := '&Accelerate';
  Label1.ShowAccelChar := True;
end;
```

- 双击 TLabel 控件，使文本折行显示。

```
procedure TForm1.Label1DblClick(Sender: TObject);
begin
  Label1.Caption := '我是中国人,我要为祖国的繁荣做贡献';
  Label1.WordWrap := True;
  Label1.Width := 80;
  Label1.Height := 50;
end;
```

3.6 文本类控件

3.6.1 引入文本类控件

在 Windows 应用程序中，用户经常与程序交互，向程序中写入数据。文本编辑控件提供了用户与程序交互的窗口，通过文本编辑控件，用户可以向程序中输入数据以进行相关操作。在 Delphi 中，文本编辑控件的基本功能被封装在 TCustomEdit 抽象基类中，然后由 TCustomEdit 派生出不同的编辑控件。

3.6.2 文本类控件属性设置

- AutoSelect

当编辑控件获得焦点时，编辑控件中的文本是否自动选中。

- CharCase

该属性确定控件中的文本大小写。

- HideSelection

确定控件失去焦点时，控件中选定的文本是否保持选中状态。

- MaxLength

确定控件包含文本的最大长度。

- PasswordChar

该属性用指定的字符隐藏实际的文本，多用于口令输入。

- ReadOnly

确定编辑框中的文本是否可以编辑。

- Text

该属性用来访问编辑控件中的文本。

- OnChange

当编辑编辑框中的文本时触发该事件。

- OnKeyPress

当编辑框获得焦点时，按任意键触发该事件。

3.6.3 文本类控件实例应用

- 双击编辑框控件，对其文本进行大小写转换。

```
procedure TForm1.Edit1DblClick(Sender: TObject);
begin
if Edit1.CharCase <> ecLowerCase then
      Edit1.CharCase := ecLowerCase
   else
      Edit1.CharCase := ecUpperCase;
end;
```

- 在编辑框中只允许输入数字。

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var
   i: Boolean;
begin
   i := (Key<#8)or(Key>#8)and(Key<#48)or(Key>#57) ;
   if i then
      Key := #0;
end;
```

3.7 按钮类控件

3.7.1 按钮类控件的主要应用

按钮是 Windows 中的常用元素，它主要用来完成一项操作。Delphi 7 为开发人员提供了丰富的按钮控件，每个按钮控件都有自己的特点和不同的用途。例如，TButton 定义了典型按钮的基本功能，TBitBtn 在继承 Tbutton 的基础上，提供了位图显示的功能，而 TSpeedButton 继承于图形控件（TGraphicControl），不具有句柄，占用系统资源较少。

3.7.2 按钮类控件属性设置

1. TButton 按钮控件

TButton 组件是按钮组件（TButtonControl）的主要派生类，它为 Delphi 程序员提供了命令按钮的基本功能。命令按钮在 Windows 程序中很常见，它通常是长方形，包含一个标题，用于响应某个操作。其主要属性如下。

- Caption

该属性用来设置按钮的标题。

- Enabled

该属性确定按钮是否可用。

- Hint

当鼠标停在按钮上时，该属性确定出现的提示信息。要想使该属性可用，需要将“ShowHint”属性设置为“True”。

- Visible

该属性确定按钮是否可用。

- OnClick

当用户单击按钮时触发该事件。

- OnEnter

当按钮获得焦点时触发该事件。

- OnExit

当按钮失去焦点时触发该事件。

- OnMouseDown

当用户按下鼠标时触发该事件。

- OnMouseMove

当用户移动鼠标时触发该事件。

- OnMouseUp

当鼠标释放时触发该事件。

2. TBitBtn 位图按钮控件

TBitBtn 是 TButton 的派生类，它继承了 TButton 的所有功能，并且提供了显示图标的能力。图标是一个小的位图图形，一般是 16 像素×16 像素，它显示在 TBitBtn 按钮的表面。其主要属性如下。

- Cursor

该属性确定鼠标在按钮上时的形状。

- Font

该属性用来设置位图按钮的字体。

- Glyph

该属性用来设置位图按钮的图标，双击该属性时会打开图片编辑器，通过图片编辑器用户可以将位图显示在位图按钮中。

- Kind

该属性确定位图按钮的种类，共有 11 个属性值，分别为“bkCustom”、“bkOK”、“bkCancel”“bkHelp”、“bkYes”、“bkNo”、“bkClose”、“bkAbort”、“bkRetry”、“bkIgnore”和“bkAll”。

 - bkCustom

 使用“Glyph”属性确定出现在位图按钮上的图形，它没有相应的“ModalResult”值。

 - bkOK

 在位图按钮的表面会出现一个绿色的“✔”标记，按钮的标题为“OK”，“ModalResult”值为“mrOK”。

 - bkCancel

 在位图按钮的表面会出现一个红色的“✘”标记，按钮的标题为“Cancel”，“ModalResult”值为“mrCancel”。

 - bkYes

 在位图按钮的表面会出现一个绿色的“✔”标记，按钮的标题为“Yes”，“ModalResult”

明日科技

值为“mrYes”。

- bkNo

在位图按钮的表面会出现一个红色的“”标记，按钮的标题为“No”，“ModalResult”值为“mrNo”。

- bkHelp

在位图按钮的表面会出现一个蓝色的“”标记，按钮的标题为“Help”。

- bkClose

在位图按钮的表面会出现一个“”标记，按钮的标题为“Close”。

- bkAbort

在位图按钮的表面会出现一个红色的“”标记，按钮的标题为“Abort”，“ModalResult”值为“mrNo”。

- bkRetry

在位图按钮的表面会出现一个绿色的“”标记，按钮的标题为“Retry”。

- bkIgnore

在位图按钮的表面会出现一个绿色的“”标记，按钮的标题为“Ignore”。

- bkAll

在位图按钮的表面会出现一个绿色的“”标记，按钮的标题为“All”。

- Spacing

确定图标与标题的距离。

- Layout

确定图标与标题怎样在按钮上排列，共有 4 个属性值，分别为“blGlyphLeft”、“blGlyphRight”、“blGlyphTop”和“blGlyphBottom”。

- blGlyphLeft

图像出现在标题的左边。

- blGlyphRight

图像出现在标题的右边。

- blGlyphTop

图像出现在标题的上边。

- blGlyphBottom

图像出现在标题的下边。

- Margin

确定按钮图像与边界的距离。

3. TSpeedButton 加速按钮控件

TSpeedButton 是非窗口控件，它不直接封装 Windows API 中的内容，不具有焦点，占用资源较少。使用 TSpeedButton 可以得到与 TBitBtn 类似的具有图标和标题的按钮。其主要属性如下。

- Constraints

该属性确定控件的大小，即确定控件宽度和高度的最大值与最小值。

- Down

标识加速按钮的状态，即凹下状态或凸起状态。该属性只有在“GroupIndex”属性不

为0时才可用。

- Flat

该属性确定加速按钮是否显示三维边框。

- Glyph

该属性可以向加速按钮添加图标。

3.7.3 按钮类控件应用

- 单击一个按钮使本身不可用，使另一个按钮可用。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Enabled := False;
  Button2.Enabled := True;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  Button1.Enabled := True;
  Button2.Enabled := False;
end;
```

- 鼠标在位图按钮上移动时更改鼠标形状和字体颜色。

```
procedure TForm1.BitBtn1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  BitBtn1.Font.Color := clBlue;
  BitBtn1.Cursor := crHandPoint;
end;
```

- 鼠标按下时改变字体颜色，鼠标释放时恢复字体颜色。

```
procedure TForm1.SpeedButton1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  SpeedButton1.Caption := 'FontChange';
  SpeedButton1.Font.Color := clBlue;
end;
procedure TForm1.SpeedButton1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  SpeedButton1.Font.Color := clBlack;
end;
```

3.8 选择类控件

3.8.1 复选框与单选按钮

TCheckBox实现了Win32复选框的功能，用于确认可选项是否被选中。在一个窗口或

组合框中可能有多个复选框，但它们之间没有任何联系，每一个复选框的选中与非选中状态都不会影响其他复选框。

TRadioButton 封装了 Windows 选项按钮的功能，也被称为单选按钮。单选按钮的特点是同一组中的单选按钮是互斥的，即一组中的一个单选按钮被选中，其他单选按钮将处于非选中状态。单选按钮是按它的父亲进行分组的，即同一个父亲的单选按钮被认为是一组。

1. TCheckBox 控件的常用属性

- AllowGrayed

该属性确定复选框是否允许处于灰色状态。如果该属性为“True”，表示复选框有 3 种状态，即选中状态、非选中状态和灰色状态，如果该属性为“False”，复选框只有两种状态，即选中状态与非选中状态。

- Checked

该属性确定复选框是否被选中。

- Enabled

该属性确定复选框是否可用。

- State

该属性确定复选框的状态。

2. TRadioButton 控件常用属性、方法

- Checked

该属性确定单选按钮是否处于选中状态。

- Caption

该属性确定单选按钮的标题。

- Hide

使单选按钮不可见。

3. TCheckBox 控件与 TRadioButton 控件属性应用

- 如果复选框被选中，将其标题添加到备注组件中。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Clear;
  if CheckBox1.Checked then
    Memo1.Lines.Add(CheckBox1.Caption+'被选中');
  if CheckBox2.Checked then
    Memo1.Lines.Add(CheckBox2.Caption+'被选中');
  if CheckBox3.Checked then
    Memo1.Lines.Add(CheckBox3.Caption+'被选中');
end;
```

- 使用按钮控制复选框状态。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
```

明日科技

```
  if CheckBox1.Checked then
     CheckBox1.Checked := False
  else
    CheckBox1.Checked := True;
end;
```

- 单击按钮，将被选中的单选按钮标题添加到备注组件中。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Clear;
  if RadioButton1.Checked then
    Memo1.Lines.Add(RadioButton1.Caption +'被选中')
  else if RadioButton2.Checked then
    Memo1.Lines.Add(RadioButton2.Caption +'被选中')
  else if RadioButton3.Checked then
    Memo1.Lines.Add(Radiobutton3.Caption +'被选中');
end;
```

3.8.2 列表框控件应用

列表框为用户提供了一组选项列表，用户可以在列表中同时选择多个选项，并且能够对这些选项排序。列表框与其他集合组件相比，便于添加和删除选项。在 Delphi 中，TListBox 控件实现了 Windows 列表框的功能。

1. TListBox 控件的常用属性

- BorderStyle

确定列表框控件是否有边框。

- ExtendedSelect

确定用户在列表框中是否能够选择一个区域。该选项只有在“MultiSelect”属性为“True”时起作用。在“ExtendedSelect”为“True”时，用户按住 Shift 键并单击选项，可以将两个选项及其之间的选项都选中，如果按 Ctrl 键并单击选项，在保持以前选项被选中的情况下，将当前选项选中。

- IntegralHeight

该属性确定是否允许列表框中的项目部分被显示。该属性为“True”时，不允许列表框中的项目被部分显示，该属性为“False”时，列表框的高度不受限制，但是列表框中最后的项目可能不会被全部显示。当“Style”属性为“lbOwnerDrawVariable”时，设置“IntegralHeight”属性没有任何影响。

- Sorted

该属性确定列表框中的选项是否按字母顺序排列。

- MultiSelect

该属性确定列表框中的项目是否允许多项同时被选中。

- ItemIndex

该属性是指选项在列表中的位置值。在列表中，第一项的位置值为 0，第二项为 1，依次类推。

❑ Items

该属性确定出现在列表中的字符串。

2. TListBox 控件应用

❑ 单击按钮，向列表框中添加选项。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ListBox1.Items.add('添加选项');
end;
```

❑ 单击按钮，对列表框中的选项按字母顺序排序。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ListBox1.Sorted := True;
end;
```

3.8.3 组合框控件应用

在 Delphi 中，TComboBox 封装了 Windows 组合框的功能。组合框由一个编辑框和一个列表框组成，主要是用来让用户在列表框中选择数据或在编辑框中输入数据，特点是占用的屏幕空间少。

1. TComboBox 控件的常用属性

❑ AutoCloseUp

该属性确定在组合框中选择一个选项，下拉列表是否自动关闭。

❑ AutoComplete

该属性确定在编辑框中输入一个字符是否将与该字符匹配的选项显示出来。

❑ AutoDropDown

该属性确定组合框获得焦点时，用户按键，下拉列表是否自动显示。

❑ DropDownCount

该属性确定下拉列表中一次可显示的字符串个数。如果列表中的选项比该属性值大，下拉列表会出现滚动条，如果列表中的项目比该属性值小，下拉列表会收缩来容纳选项。

❑ ItemIndex

该属性指明当前项在组合框中的位置。如果组合框中没有选项，该值为-1，否则列表框中第一项的“ItemIndex”值为 0，第二项为 1，依次类推。

❑ MaxLength

该属性限制用户在编辑框中输入的字符数，如果“MaxLength”为 0，表示没有限制，可以在编辑框中输入任意多个字符，否则在编辑框中输入的字符不能多于“MaxLength”属性值。

❑ Text

该属性确定编辑框中的文本。

2. TComboBox 控件属性应用

❑ 单击按钮，读取当前选项在组合框中的位置。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Caption := IntToStr(ComboBox1.ItemIndex);
end;
```

- 单击按钮，限制在编辑框中输入的字符数。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ComboBox1.MaxLength := 5;
end;
```

3.9 时钟控件

3.9.1 时钟控件主要属性、事件

时钟控件允许用户在指定时间间隔内触发事件，尽管时间间隔以毫秒指定，但时间控件只是近似的，并且受到系统性能影响。其主要属性、方法如下。

- Enabled

确定时钟控件是否可用。

- Interval

确定触发 OnTimer 事件的时间间隔。

- OnTimer

该事件在指定的时间间隔过后触发。

3.9.2 时钟控件在用户界面中的应用

- 更改窗体的标题。

```
var
  Form1: TForm1;
  i: Integer = 0;
implementation
{$R *.dfm}
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Caption := IntToStr(i);
  Inc(i);
end;
```

- 制作流动字幕。

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if Label1.Left >0 then
    Label1.Left := Label1.Left-2
  else
    Label1.Left := Form1.Width;
end;
```

3.10 框架控件

3.10.1 框架控件在用户界面中的应用

框架控件 TFrame 是组件的容器，类似于窗体，能够被嵌套在其他的窗体或框架内。在使用框架控件前，需要创建一个框架单元，可以通过单击系统菜单“File”/“New”/“Frame”向工程中添加一个框架单元。

3.10.2 框架控件主要属性、方法

- AutoScroll

该属性为 True 时，滚动条会在需要时自动显示。

- ParentBackground

该属性确定控件是否使用父类的主题背景。

- ShowHint

该属性确定鼠标在控件上移动时，是否显示 Hint 属性的提示信息。

- Hint

该属性确定鼠标在控件上移动时显示的提示信息，为了使工具提示显示 Hint 属性值，需要将控件的 ShowHint 属性置为 True。

- Hide

该方法使控件不可见。

- Parent

该属性可以获得或设置框架的父类。

- DisableAutoRange

该方法禁止自动滚动，当在框架控件中重新排列控件时，使用该方法可以临时禁止自动滚动，当完成框架控件的排列时，可以使用 EnableAutoRange 重新激活自动滚动。

3.10.3 框架控件实例应用

- 利用框架实现界面与代码重用。

```
//框架单元代码，在框架中放置一个编辑框控件
//编辑框中不允许输入非法字符
procedure TFrame2.Edit1KeyPress(Sender: TObject; var Key:
Char);
var
  i: Boolean;
begin
  i := (Key<#8)or(Key>#8)and(Key<#48)or(Key>#57);
  if i then
    Key := #0;
end;
```

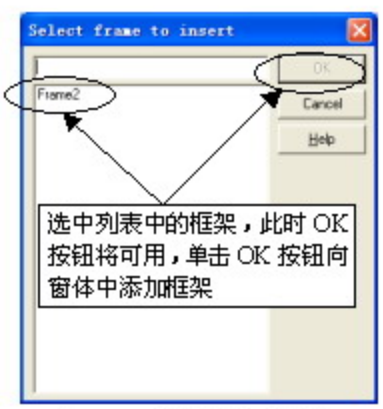


图 3.10 选择框架对话框

在主窗体中放置一个 TFrame 组件，会弹出如图 3.10 所示的对话框。

在列表中选中框架，单击 OK 按钮向主窗体中添加框架，会发现框架控件中包含了一个编辑框控件，运行程序，在编辑框中只允许输入数字。

3.11 修饰类控件

3.11.1 主要修饰类控件

在 Delphi 的图形控件中提供了一些修饰类控件，这些控件在开发程序过程中并没有太大用处，只是用来修饰程序界面，使界面更美观。常用的修饰类控件主要有 TBevel、TShape 等。TBevel 控件可以在窗体中绘制框架和线条，它会呈现一种真实的三维效果，将它放在其他控件的上面，会使控件从视觉上加以分离，使控件看起来更美观。Tshape 控件可以在窗体中显示简单的几何图像。

1. TShape 控件常用属性

- BoundRect

确定控件占用的边界矩形。

- Brush

确定用于填充形体对象的颜色和模式。

- Pen

确定用于形体对象的画布上绘制线条或轮廓的笔的种类。

- Shape

确定形体对象的形状，共有 6 个属性值，其意义如下。

 - stCircle

 形体对象为圆形。

 - stEllipse

 形体对象为椭圆形。

 - stRectangle

 形体对象为矩形。

 - stRoundRect

 形体对象为圆角矩形。

 - stRoundSquare

 形体对象为圆角正方形。

 - stSquare

 形体对象为正方形。

- Visible

确定形体对象是否可见。

2. TBevel 控件常用属性

- Shape

确定 TBevel 组件的形状，共有 7 个属性值，意义如下。

 - bsBottomLine

 斜体对象在其区域内只显示底部的一条线。

 - bsBox

 斜体对象在其区域内显示一个方框形。

明日科技

- bsFrame

斜体对象在其区域内显示一个框架形。

- bsLeftLine

斜体对象在其区域内只显示左边的一条线。

- bsRightLine

斜体对象在其区域内只显示右边的一条线。

- bsTopLine

斜体对象在其区域内只显示顶部的一条线。

- bsSpacer

斜体对象显示一个空白区域。

- Style

该属性确定斜体对象的风格，“bsRaised”表示凸起，“bsLowered”表示下凹。

- Top

确定斜体对象与父类控件的上边距。

- Visible

确定斜体对象是否可见。

3.11.2 使用修饰类控件美化用户界面

在设计程序界面时，适当地使用修饰类组件可以使界面增色不少，下面是一个商品销售管理的程序界面，在界面中的销售票号与销售时间下面作者使用了 TShape 控件，使销售票号和销售时间变得更醒目，在标题部分，作者使用了 TBevel 控件，使标题呈现出三维效果。程序界面如图 3.11 所示。



图 3.11 销售管理程序界面

第 4 章 菜单栏、工具栏和对话框

4.1 菜单程设计

4.1.1 创建菜单与菜单项

在设计 Windows 应用程序时，菜单几乎是不可缺少的组件，通过菜单，用户可以调用系统的各个功能。在 Delphi 中，菜单被封装在 TMainMenu（主菜单）类与 TPopupMenu（弹出式菜单）类中，程序员可以通过菜单设计器来设计菜单，双击 TMainMenu 组件或 TPopupMenu 组件可以打开菜单设计器，用户可以通过在对象观察器中修改菜单项的“Caption”属性设置菜单项，如图 4.1 所示。

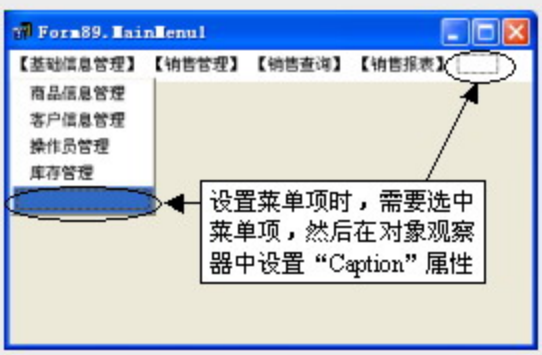


图 4.1 设置菜单项

进入菜单设计器后，会发现有一个空的菜单项。菜单项是出现在菜单中的选项，它是由 TMenuItem 类派生出来的对象，每个菜单项都有自己的属性、方法和事件，当用户单击一个菜单项时，会触发该菜单项的“OnClick”事件。

4.1.2 在菜单中添加分隔线

首先双击 TMainMenu 组件打开菜单设计器，然后选中菜单项，在对象观察器中设置菜单项的“Caption”属性为“-”，如图 4.2 所示。

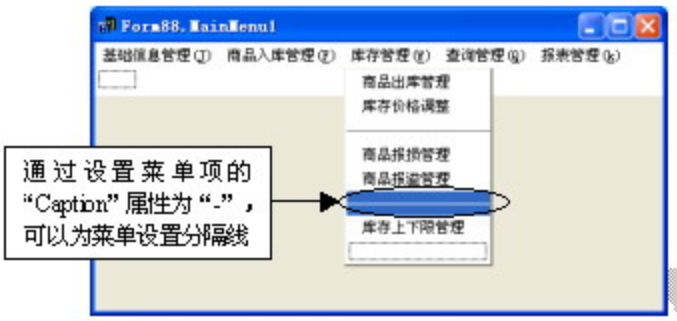


图 4.2 添加分隔线

4.1.3 设计级联菜单

在设计菜单时，有时菜单项需要包含许多子菜单，而每个子菜单又需要多个子菜单，这种菜单称为级联式菜单。设计级联式菜单，可以在菜单设计器中选中一个菜单项，然后用鼠标右键单击该菜单项，会弹出一个快捷菜单，如图 4.3 所示。

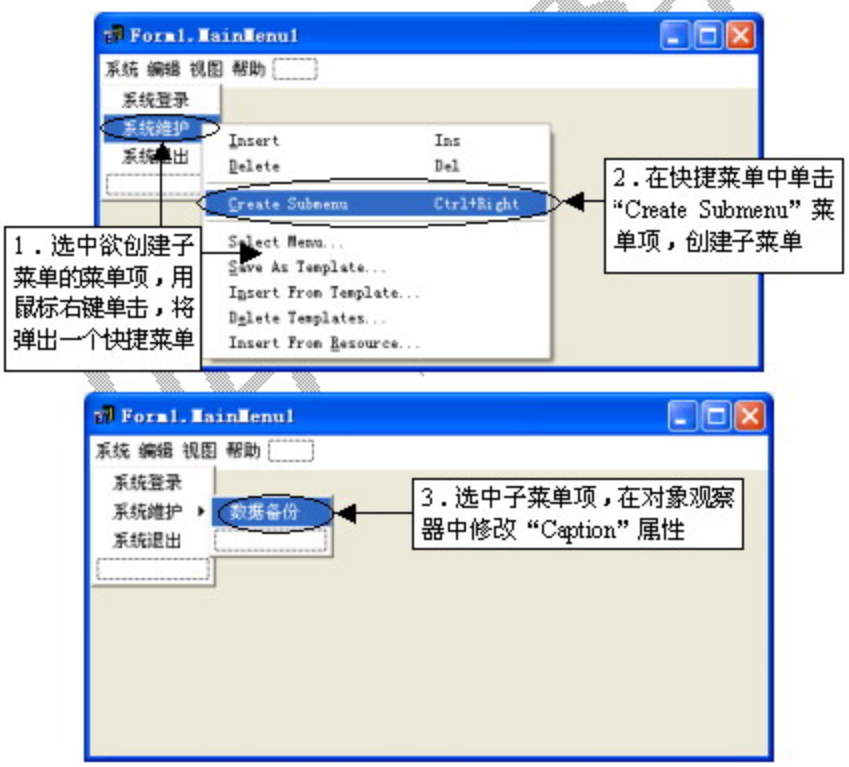


图 4.3 级联菜单

4.1.4 设计单选菜单

单选式菜单是对于一组单选菜单项，应用程序一次只能选中其中的一个。要设计单选

式菜单，可以将一组菜单中的各个菜单项的“RadioItem”属性设置为“True”，然后将其“GroupIndex”属性设置为相同的属性值，这样，当一个菜单项的“Checked”属性为“True”时，其他菜单项的“Checked”属性便会为“False”，如图 4.4 所示。

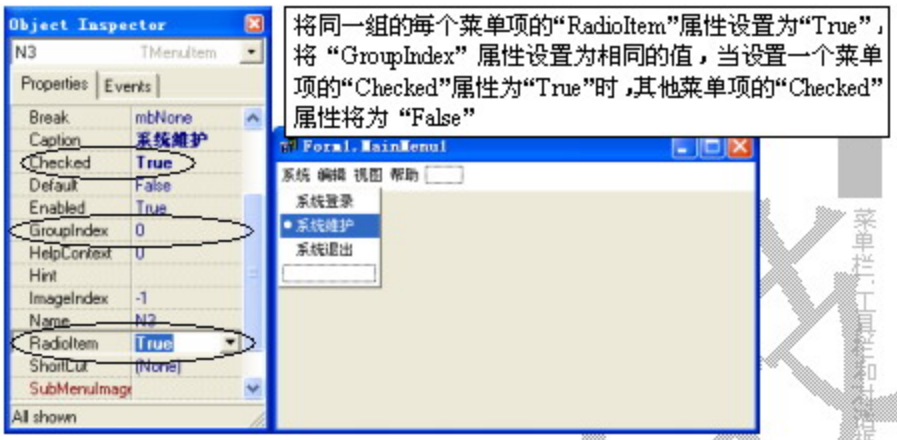


图 4.4　单选式菜单

4.1.5　设计复选菜单

复选式菜单是对于一组菜单项，可以有多个菜单项同时被选中。设置复选式菜单需要将各个菜单项的“RadioItem”属性设置为“False”，设置“Checked”属性为“True”，菜单项会有复选标记，反之没有，如图 4.5 所示。



图 4.5　复选式菜单

4.1.6　为菜单添加图标

在 Delphi 中为菜单添加图标需要使用 TImageList 组件存储图标，然后利用菜单组件的一些属性将 TImageList 组件中的图标显示在菜单中。

（1）在窗体中放置一个 TImageList 组件，双击该组件打开图像列表编辑器，如图 4.6 所示。

（2）选中菜单组件，在对象观察器中设置其“Images”属性为“ImageList1”，在菜单设计器中选中菜单项，在对象观察器中单击“ImageIndex”属性，从弹出的下拉列表中选择一个图标索引，具体操作如图 4.7 所示。

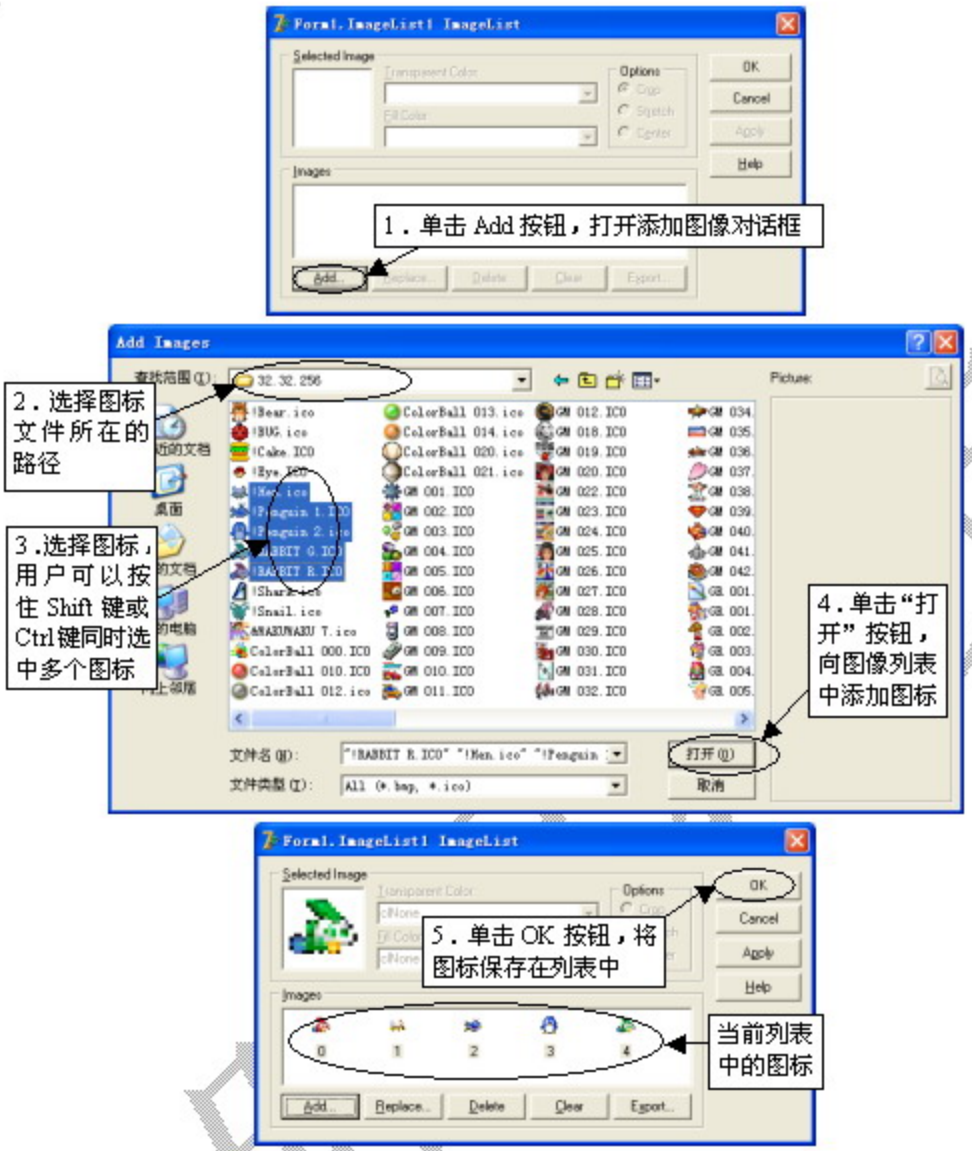


图 4.6　图像列表



图 4.7　图标菜单

4.1.7 菜单命令的响应

设计菜单的目的是单击菜单项能够执行一些操作。前面已经介绍过，菜单项有自己的属性、方法和事件，当用户单击菜单项时，会响应菜单项的“OnClick”事件，所以可以在菜单项的“OnClick”事件中添加事件处理过程。开发人员可以在菜单设计器中双击菜单项或者在窗体中单击菜单项，在代码编辑器中会生成如下事件处理过程：

```
procedure TForm1.N2Click(Sender: TObject);
begin


end;
```

在该事件处理过程中添加实现代码以响应菜单项的单击事件。

4.1.8 动态管理菜单

在开发应用程序时，有时需要在程序运行时动态管理菜单。例如，在设计复选菜单时，当用户单击一次菜单项时，会出现复选标记，当用户再次单击该菜单项，复选标记应该消失。由于每一个菜单项都是一个独立的对象，所以在程序中只要使用一些代码设置菜单项的属性或者调用菜单项的方法便可以实现菜单的动态管理。菜单设计如图 4.8 所示。



图 4.8 动态管理菜单

以下是设置复选菜单的程序代码：

```
procedure TForm1.N5Click(Sender: TObject);
begin
  if N5.Checked = False then
  begin
    Caption := 'Checked';
    N5.Checked := True;
  end
  else
  begin
    Caption := 'unChecked';
    N5.Checked := False;
  end;
end;
```

4.1.9 设计弹出式菜单

在 Windows 操作系统中，用户随意用鼠标右键单击桌面、文件夹、状态栏等都会弹出

一个快捷菜单，被称之为弹出式菜单。在 Delphi 中，弹出式菜单被封装在 TPopupMenu 类中。弹出式菜单的设计与主菜单设计相同，是通过菜单设计器来添加菜单项，在此，笔者就不做过多的介绍了，只是在设计完菜单项后，需要将一个控件的“PopupMenu”属性指向弹出式菜单，运行程序时，当用户用鼠标右键单击该控件时便会弹出一个快捷菜单，其中包含了我们所设计的菜单项。弹出式菜单设置如图 4.9 所示。

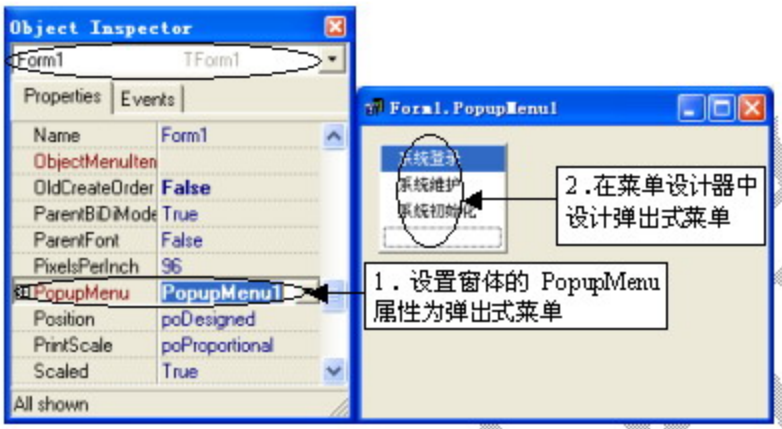


图 4.9　弹出式菜单设计

程序运行时，用鼠标右键单击窗体，将弹出所设计的菜单，如图 4.10 所示。

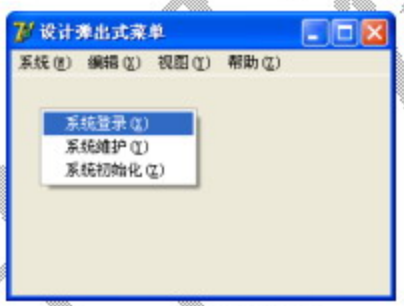


图 4.10　弹出式菜单效果图

4.2　设计工具栏

4.2.1　TToolBar 组件与 TToolButton 组件

工具栏组件（TToolBar）位于系统菜单的下方，它提供了存放工具栏按钮（TToolButton）的场所。在开发应用程序时，经常将一些常用的命令放在工具栏中，这已经成为一种惯例，不过这给用户确实带来了方便，因为使用一些命令不需要在级联的菜单中一层层查找，只需单击工具栏上的按钮便可以响应菜单命令。

在 Delphi 中，向工具栏中添加工具栏按钮是非常方便的，用鼠标右键单击工具栏，在弹出的快捷菜单中单击“New Button”菜单项，会发现工具栏中存在了一个工具栏按钮。在工具栏中选中工具栏按钮，在对象观察器中会看到它是一个独立的对象，即 TToolButton 类的对象，有自己的属性、方法和事件。

4.2.2 设置按钮样式

1. 向工具栏按钮中添加图标

向工具栏按钮中添加图标，首先需要使用 TImageList 组件存储图标，向 TImageList 组件中添加图标的具体操作请参考 4.1.6 小节，然后设置工具栏的“Images”属性为 TImageList 组件，此时会发现工具栏按钮中显示了 TImageList 组件中的图标，但是工具栏按钮中的图标顺序与 TImageList 组件中图标顺序相同，如果要调整工具栏按钮中的图标顺序，可以选中一个工具栏按钮，在对象观察器中重新设置“ImageIndex”属性即可。工具栏设计如图 4.11 所示。



图 4.11　工具栏设计

2. 调整工具栏按钮风格

工具栏按钮 TToolButton 有一个“Style”属性，该属性用来设置按钮的风格，共有 5 个属性值，意义如下。

- tbsButton

表示一个按钮出现在工具栏上，其功能与 TSpeedButton 组件类似。

- tbsCheck

表示单击一次按钮时，按钮会呈现凹陷状态，直到再一次单击该按钮时，按钮恢复原来的状态。

- tbsDropDown

表示在按钮旁边会出现一个下拉箭头，适合于访问下拉式菜单。

- tbsSeparator

表示按钮作为一个空白区出现在工具栏上，用于分隔其他的按钮。

- tbsDivider

表示按钮作为一个垂直线出现在工具栏上，用于分隔其他的按钮。

图 4.12 显示了 5 种不同风格的工具栏按钮。

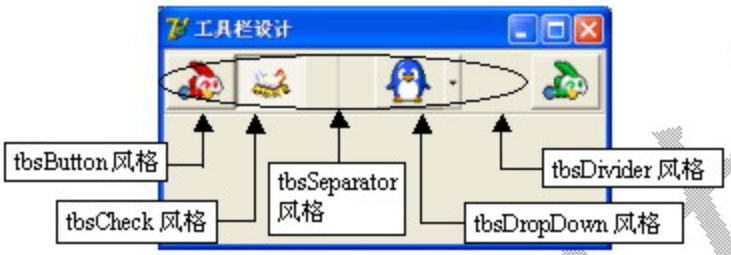


图 4.12　工具栏按钮风格

4.2.3　为工具栏设置响应代码

在前面已经说过，工具栏按钮是一个独立的对象，有其自己的属性、方法和事件。工具栏按钮常用的事件便是鼠标单击事件（OnClick）。如果用户想要单击工具栏按钮时响应某个菜单命令，可以选中工具栏按钮，在对象观察器中打开“Events”选项卡，单击“OnClick”事件，在弹出的下拉列表中选择一个菜单项的单击事件，使工具栏按钮的单击事件与菜单命令关联。如果用户想要单独处理工具栏按钮的单击事件，可以直接双击工具栏按钮，在代码编辑器中会自动生成该按钮的单击事件处理过程，在事件处理过程中输入实现代码。

4.2.4　为工具栏设置提示信息

当工具栏上的按钮比较多时，很难区分每个按钮的功能，此时有必要为按钮设置提示信息。在 Delphi 中为按钮设置提示信息是非常方便的，首先在工具栏中选中一个按钮，在对象观察器中设置其“ShowHint”属性为“True”，然后在“Hint”属性中设置提示信息，例如设置为“系统登录”，运行程序，当鼠标移动到该按钮上时，便会出现“系统登录”提示信息，如图 4.13 所示。

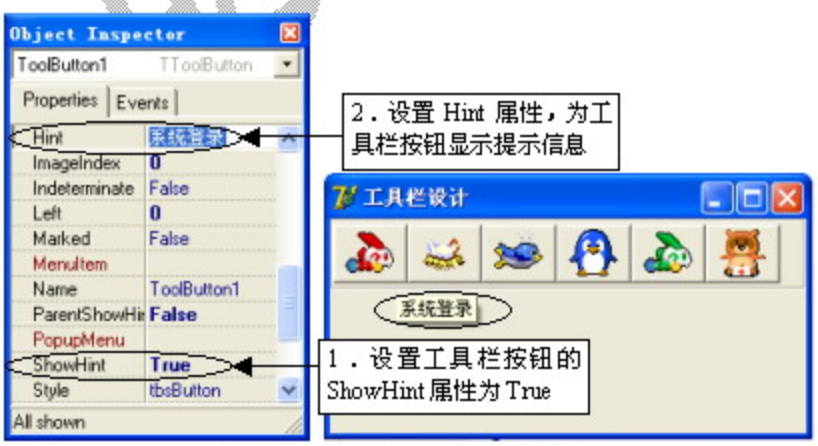


图 4.13　为工具栏按钮设置提示信息

4.3 通用对话框

4.3.1 通用对话框的引入

交互类对话框是由操作系统提供的任何应用程序都可获得的对话框，使用这些对话框，可以为用户提供他们所期望的一致性的标准界面。在 Delphi 中，这些对话框被封装在 TCommonDialog 及其派生类中，在开发程序过程中，常用的对话框组件主要有 TOpenDialog、TFontDialog、TColorDialog 和 TPrintDialog 等。

4.3.2 打开对话框和保存对话框

1. 打开对话框 TOpenDialog

- DefaultExt

确定打开对话框中文件的默认扩展名。

- FileName

确定由打开对话框返回的文件名。

- Filter

该属性指定一个文件名掩码列表，用于在打开对话框中确认对话框将显示哪些类型文件。

- FilterIndex

该属性确定当对话框打开时，哪个过滤器成为默认的过滤器。在 Delphi 的开发环境中有一个 Filter 编辑器，它可以为打开对话框定义过滤器，过滤器是一个文件名称规范集，用于确定对话框中显示的文件类型。当单击“File”/“Open”菜单时，会显示一个打开对话框，在打开对话框中使用默认的过滤器仅显示 Delphi 文件，如图 4.14 所示。

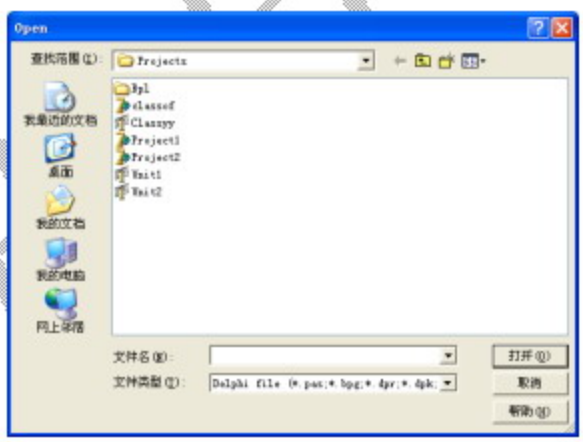


图 4.14 打开对话框

- InitialDir

确定对话框打开时使用的目录。

- Execute

该方法用于显示打开对话框。

2. 保存对话框 TSaveDialog

- DefaultExt

指定默认的文件扩展名。

- FileName

确定由保存对话框返回的文件名。

- Filter

该属性指定一个文件名掩码列表，用于在打开对话框中确认对话框将显示哪些类型文件。

- Execute

该方法显示保存对话框。

3. 属性应用

从 TOpenDialog 组件中读取打开的文件。

```
procedure TForm1.ToolButton1Click(Sender: TObject);
begin
  OpenDialog1.Execute;
  Caption := OpenDialog1.FileName;
end;
```

运行程序，单击工具栏中的按钮将调出“打开”对话框，如图 4.15 所示。

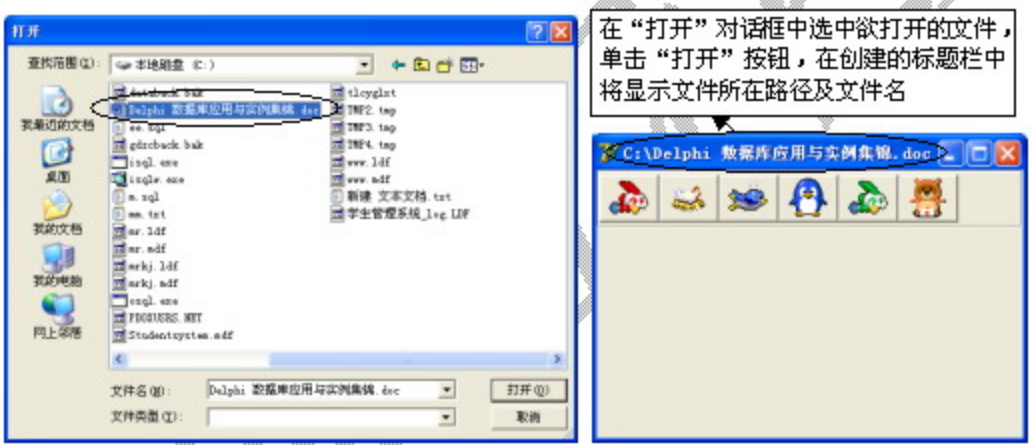


图 4.15 读取文件名

- 创建过滤器，使打开对话框只显示工程文件（.dpr）和单元文件（.pas）。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Filter := 'Delphi文件|*.dpr;*.pas';
  OpenDialog1.Execute;
end;
```

运行程序，如图 4.16 所示。

- 将数据写入指定的文件中。

```
procedure TForm1.Button1Click(Sender: TObject);
var
  FileHandle: Integer;
  FileName: String;
  Filevar: TextFile;
```

```
begin
  SaveDialog1.Filter := '文本文件|*.txt';
  if SaveDialog1.Execute then
  begin
    FileName := SaveDialog1.FileName;
    FileName := FileName + '.txt';
    if FileExists(FileName) then
      if Application.MessageBox('该文件已经存在,是否要替换?','提示',mb_YesNo)= Id_No then
        exit;
    FileHandle := Filecreate(pchar(FileName));
    FileClose(FileHandle);
    AssignFile(filevar,FileName);
    rewrite(filevar);
    write(filevar,'我是中国人');
    closeFile(filevar);
  end;
end;
```

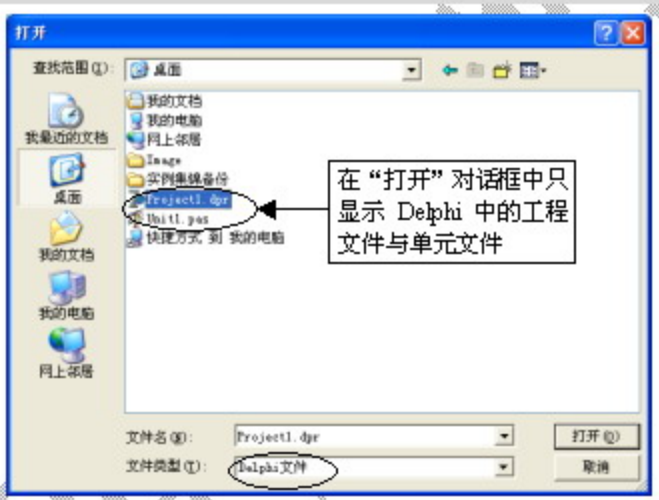


图 4.16　过滤文件

运行程序，如图 4.17 所示。

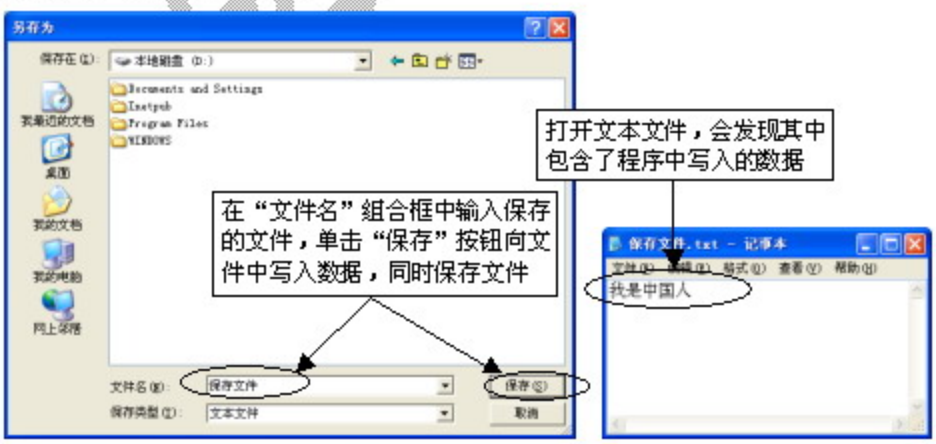


图 4.17　利用保存对话框保存文件

4.3.3 颜色对话框和字体对话框

1. 颜色对话框 TColorDialog

- Color

确定由颜色对话框返回的颜色。

- CustomColors

确定自定义颜色部分是否被显示。

- Execute

该方法会打开颜色对话框。

2. 字体对话框 TFontDialog

- Device

该属性确定由字体对话框返回的字体影响的绘制设备，共有 3 个属性值，分别为：

 - fdBoth

 影响打印机和屏幕；

 - fdPrinter

 影响打印机；

 - fdScreen

 影响屏幕。

- MaxFontSize

确定在字体对话框中可以使用的最大字体，该属性只有在“Options”选项中的“fdLimitSize”为“True”时才起作用。

- MinFontSize

确定在字体对话框中可以使用的最小字体，该属性只有在“Options”选项中的“fdLimitSize”为“True”时才起作用。

- Execute

该方法用于打开字体对话框。

3. 属性应用

- 利用颜色对话框更改窗体背景颜色。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ColorDialog1.Execute;
  Color := ColorDialog1.Color;
end;
```

运行程序，打开“颜色”对话框，为窗体设置背景色，如图 4.18 所示。

- 使用字体对话框更改编辑框字体。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  FontDialog1.Execute;
  Edit1.Font := FontDialog1.Font;
```


明日科技

```
    Edit2.Font := FontDialog1.Font;
    Edit3.Font := FontDialog1.Font;
    Edit4.Font := FontDialog1.Font;
end;
```

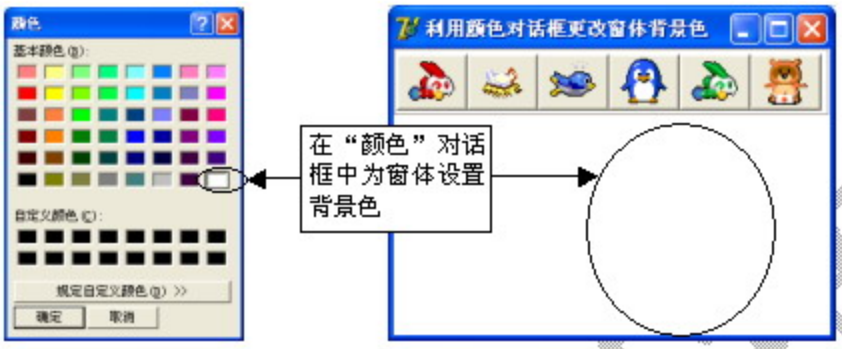


图 4.18　利用颜色对话框改变窗体背景色

运行程序，打开“字体”对话框，如图 4.19 所示。



图 4.19　利用字体对话框更改字体

4.3.4　打印对话框和替换对话框

1. 打印对话框 TPrintDialog

- Collate

确定打印工作是否自动分页。

- Copies

确定打印的份数。

- Execute

该方法显示打印对话框。

- FromPage

确定打印的起始范围。

- MaxPage

确定允许用户打印的最大页号。

- MinPage

确定允许用户打印的最小页号。

- PrintRange

确定打印范围。

- PrintToFile

确定打印工作是否打印到文件。

- ToPage

确定打印范围的终止部分。

2. 替换对话框 TReplaceDialog

- FindText

确定要查找的文本。

- ReplaceText

确定替换查找文本的文字。

- OnReplace

当发生替换时触发该事件。

- OnFind

当发生查找时触发该事件。

3. 属性应用

利用打印对话框设置打印机。

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Page: Integer;
begin
  PrintDialog1.FromPage := 1;
  PrintDialog1.MinPage := 1;
  PrintDialog1.Copies := 2;
  PrintDialog1.Execute;
  with Printer do
  begin
    BeginDoc;
    For Page :=0 to PageControl1.PageCount-1 do
    begin
      PageControl1.Pages[Page].PaintTo(Handle,5,5);
      NewPage;
    end;
    EndDoc;
  end;
end;
```


北大科技

运行程序，调出“打印”设置对话框，如图 4.20 所示。

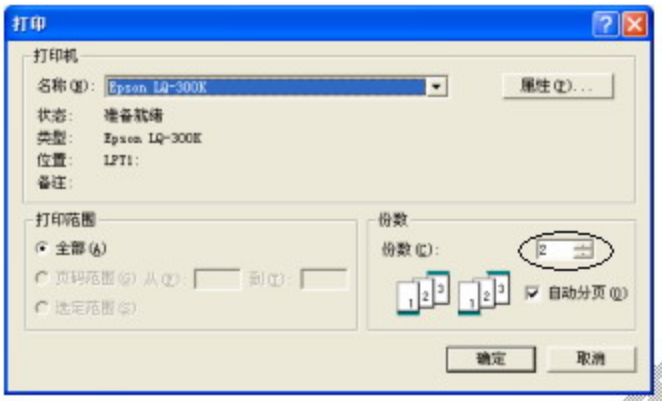


图 4.20 “打印”对话框

❑ 使用 TReplaceDialog 组件查找文本并替换文本。

```
procedure TForm1.ReplaceDialog1Replace(Sender: TObject);
var
   FoundAt: Longint;
   BeginPos,EndPos: Integer;
begin
   BeginPos := 0;
   EndPos := Length(RichEdit1.Text);
   FoundAt := 0;
   while FoundAt <>-1 do
   begin
      FoundAt := RichEdit1.FindText(Trim(ReplaceDialog1.FindText),BeginPos,EndPos,
[stMatchCase]);
      if FoundAt<>-1 then
      begin
         RichEdit1.SetFocus;
         RichEdit1.SelStart := FoundAt;
         RichEdit1.SelLength := Length(Trim(ReplaceDialog1.FindText));
         BeginPos := FoundAt + Length(Trim(ReplaceDialog1.ReplaceText));
         RichEdit1.SelText := Trim(ReplaceDialog1.ReplaceText);
      end;
   end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
   ReplaceDialog1.Execute;
end;
```

运行程序，调出“替换”对话框，具体操作如图 4.21 所示。

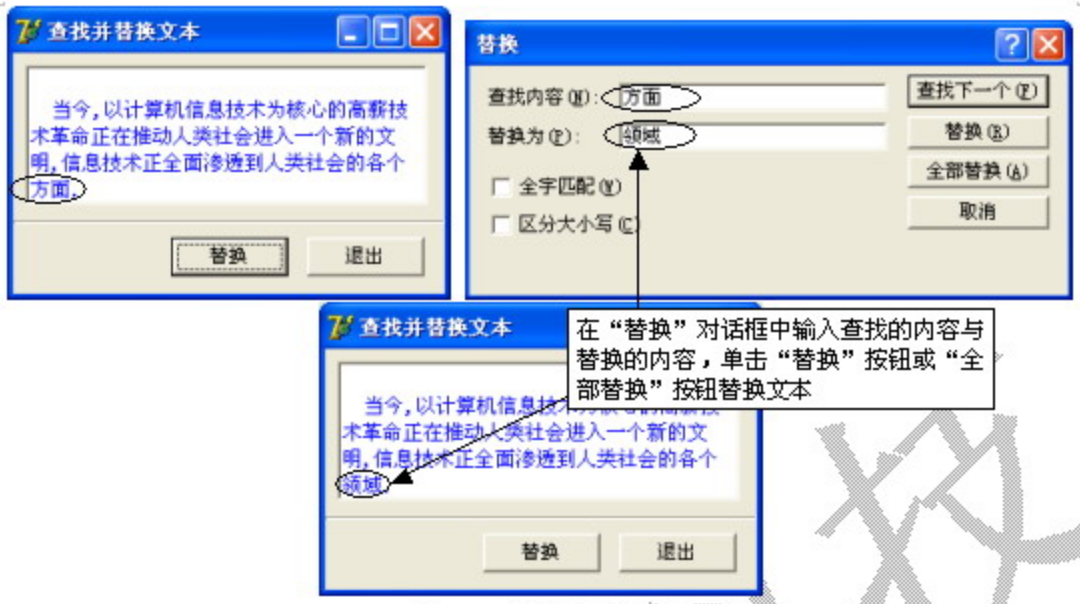


图 4.21　查找并替换文本

明日科技

第5章 数据库开发技术

5.1 BDE 数据库程序开发技术

5.1.1 BDE Administrator

BDE全称为Borland Database Engine，是Borland公司推出的一个数据库引擎，使用它可以连接各种数据库。Delphi数据库应用程序大部分是通过BDE来完成访问数据库的。

BDE管理器是开发人员与BDE沟通的图形窗口，通过BDE管理器，开发人员可以对BDE的参数与功能进行配置。单击系统菜单“所有程序”/“Borland Delphi 7”/“BDE Administrator”可以打开BDE管理器，如图5.1所示。

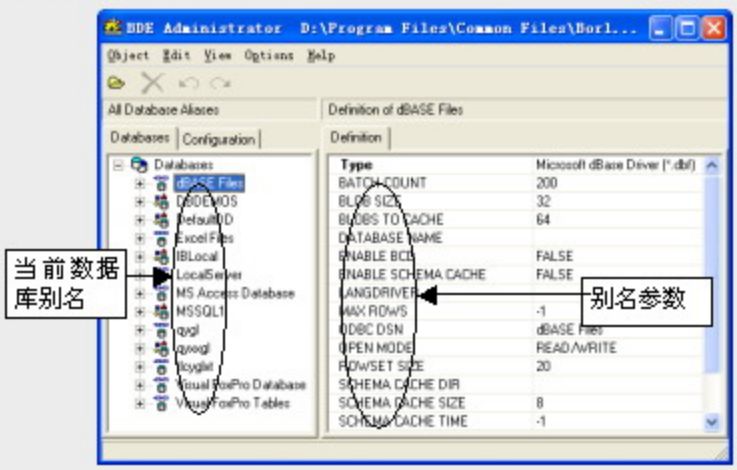


图5.1 BDE管理器

5.1.2 使用BDE连接SQL Server数据库

(1) 打开BDE管理器，如图5.1所示。用鼠标右键单击“Databases”

菜单项，会弹出一个快捷菜单，如图 5.2 所示。

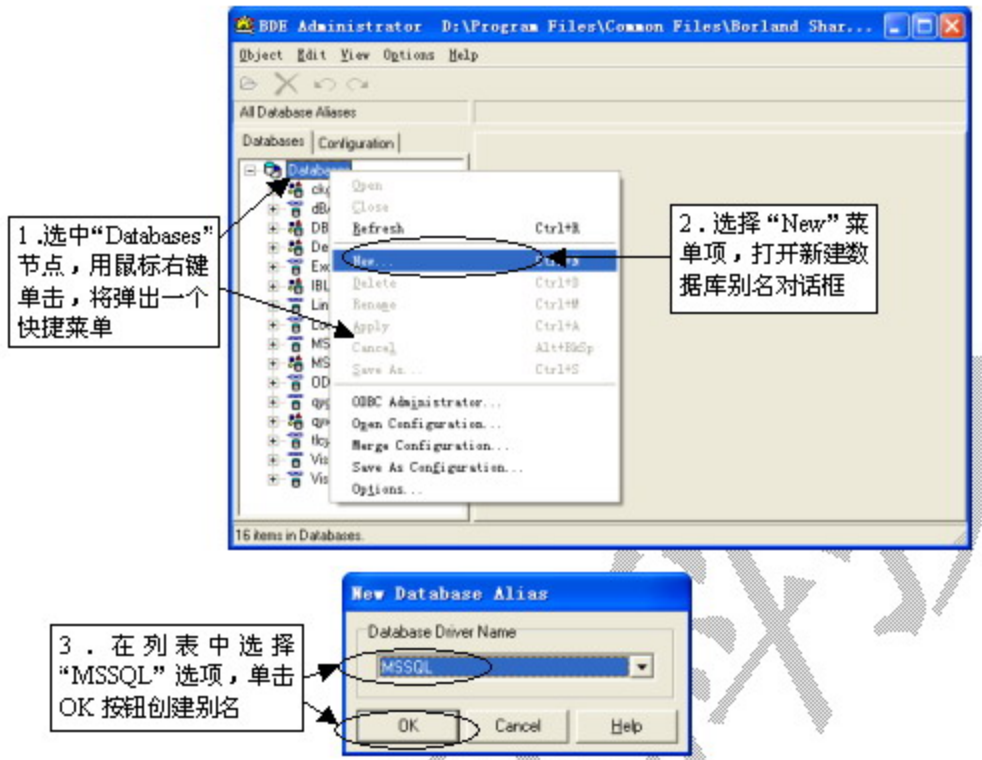


图 5.2　新建数据库别名

（2）配置数据库别名，具体操作如图 5.3 所示。

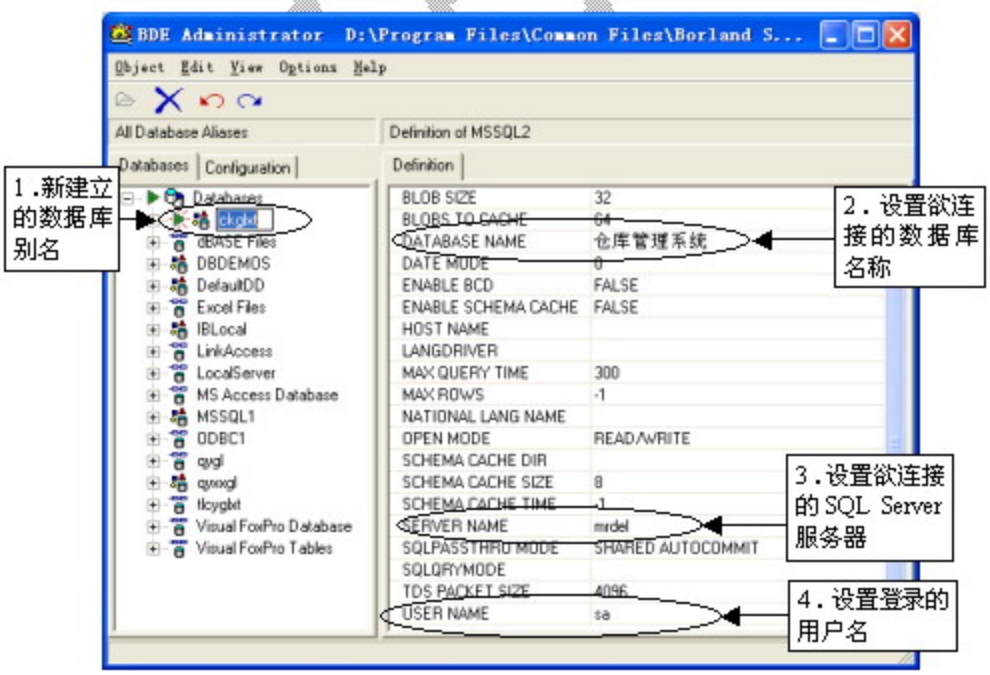


图 5.3　设置参数

（3）保存别名参数设置，连接数据库，具体操作如图 5.4 所示。

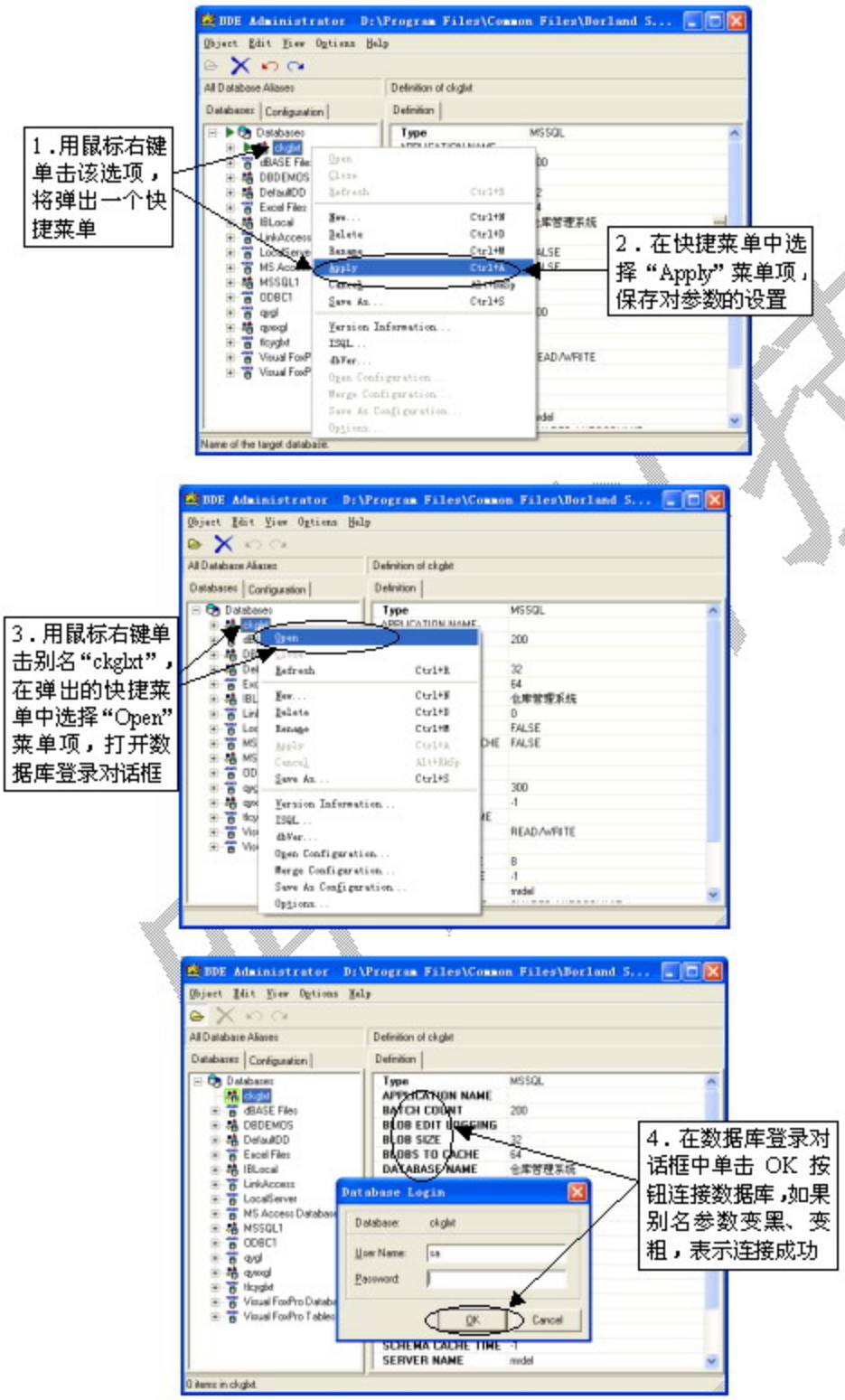


图 5.4　连接数据库

5.1.3 TTable 组件

TTable 组件主要用于访问数据表，该组件通过 BDE 连接数据库。TTable 组件不仅可以连接数据表，还可以进行简单的查找和过滤。其重要属性、方法如表 5.1 所示。

表 5.1　　TTable 组件的重要属性、方法

属性、方法	描　　述
Active	确定是否打开与数据集的连接
DatabaseName	确定该组件连接的数据库名称（BDE 中的别名）
TableName	确定该组件连接的数据表
Filtered	确定对结果集是否进行过滤
Filter	该属性用来确定过滤条件
Open	打开与数据库的连接
Close	关闭与数据库的连接

下面介绍如何使用 TTable 组件访问数据表。

（1）新建一个应用程序，在窗体中放置 TTable 组件、TDataSource 组件、TDBGrid 等组件，如图 5.5 所示。

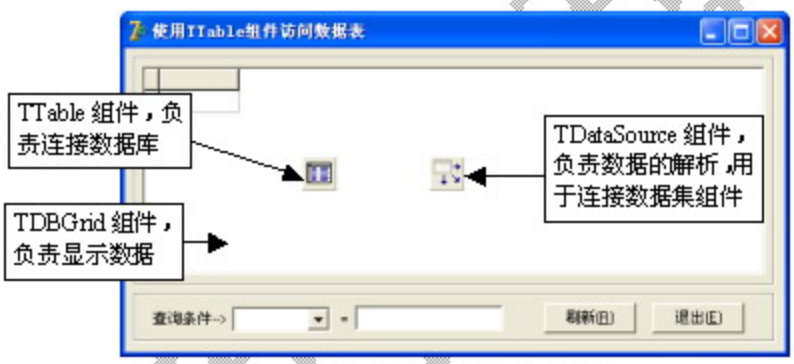


图 5.5　各组件在窗体中的放置

（2）设置各组件的属性，具体操作如图 5.6 所示。

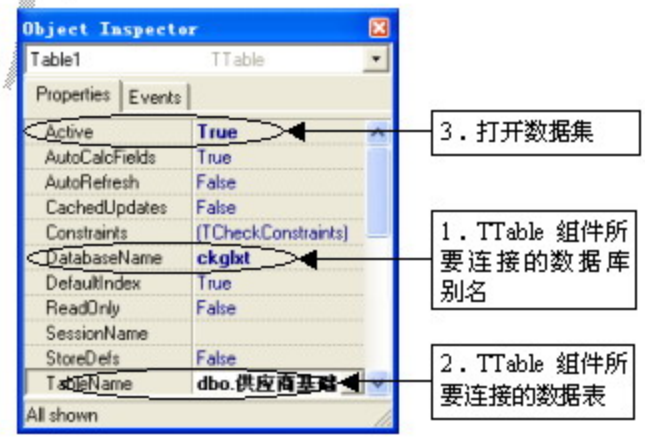




图 5.6　组件属性设置

（3）当 TTable 组件连接到数据表中时，并没有将数据显示出来，因为 TTable 组件只是建立与数据表的连接，并不具有数据显示的功能。将数据显示出来还需要进行两个步骤，首先需要使用 TDataSource 组件负责数据的解析，将 TDataSource 组件的“DataSet”属性设置为 TTable 组件，然后使用 TDBGrid 组件负责数据的显示，将 TDBGrid 组件的“DataSource”属性设置为 TDataSource 组件，此时，会发现数据已经显示在 TDBGrid 组件中，如图 5.7 所示。



图 5.7　显示数据

5.1.4　TQuery 组件

通过 TQuery 组件，用户可以使用 SQL 语句来操作数据库，所以对于一些复杂的添加、修改、删除等操作使用 TQuery 组件比 TTable 组件更加灵活。其重要属性、方法如表 5.2 所示。

表 5.2　　TQuery 组件的重要属性、方法

属性、方法	描　　述
DatabaseName	确定该组件连接的数据库名称（BDE 中的别名）
SQL	该属性用来设置 SQL 语句
Open	执行 SQL 语句，该方法要求有结果集返回，多用于 Select 语句
ExecSQL	执行 SQL 语句，该方法要求没有结果集返回，多用于 Insert、Update、Delete 等语句
RequestLive	该属性确定返回的结果集是否可以修改
Params	读取或设置 SQL 语句参数名称、参数值、参数类型
Bof	确定当前指针是否处在数据集的起始位置
Eof	确定当前指针是否处在数据集的末尾

下面介绍如何使用 TQuery 组件访问数据表。

（1）新建一个应用程序，在窗体中放置 TQuery 组件、TDBGrid 组件、TDataSource 组件，界面如图 5.8 所示。



图 5.8　使用 TQuery 组件访问数据表

（2）设置各组件的属性，具体操作如图 5.9 所示。

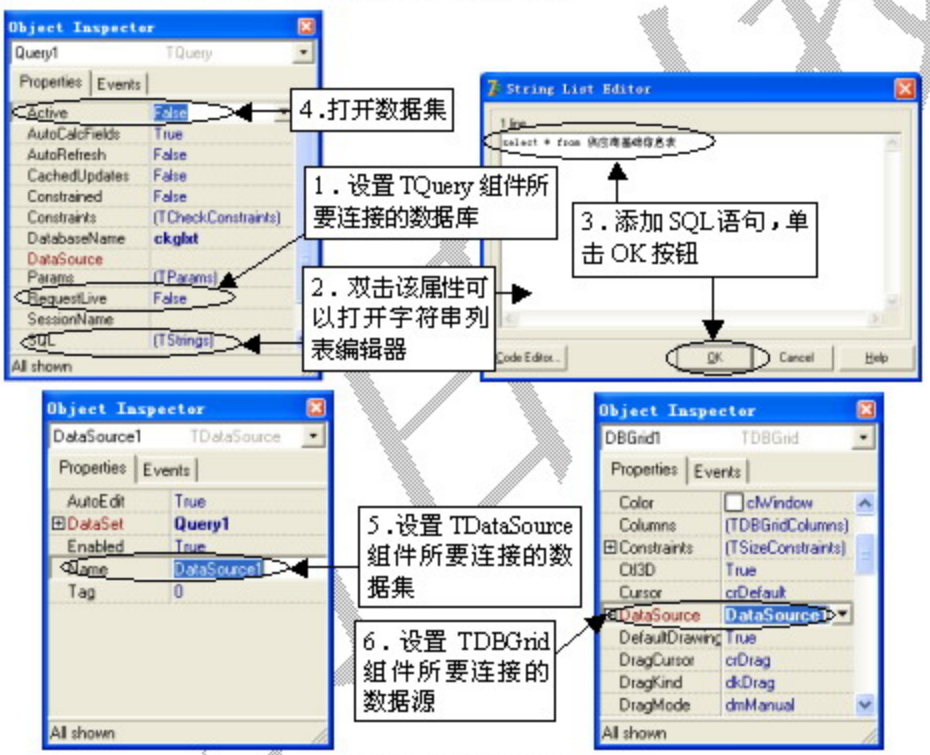


图 5.9　设置组件属性

（3）运行程序，效果如图 5.10 所示。



图 5.10　使用 TQuery 组件操作数据库

5.1.5 TDatabase 组件

TDatabase组件提供了对事务的支持。事务具有原子性，它使一组操作要么全部执行，要么一个也不执行。在开发数据库应用程序时，经常需要同时向几个表中插入数据。例如，在商品销售时，需要向销售票号表、销售表插入数据，同时还需要修改库存数量，如果在向销售表中插入数据时出错，在销售票号表中已经存在了销售记录，这显然违背了数据完整性。如果在销售过程中使用事务，则可以避免上述情况的发生，在销售商品时，如果向销售表中插入数据时出错，可以进行事务回滚，之前的操作会被取消。TDatabase组件的重要属性、方法如表 5.3 所示。

表 5.3　　TDatabase 组件的重要属性、方法

属性、方法	描　　述
AliasName	确定该组件连接的数据库名称（BDE 中的别名）
Connected	确定数据库连接是否被激活
DatabaseName	指定 TDatabase 组件提供给其他数据集组件的 DatabaseName
LoginPrompt	确定是否出现登录提示对话框
CloseDatasets	关闭所有的数据集，但不会关闭数据库的连接
StartTransaction	开始一个事务
Commit	提交事务
Rollback	回滚事务
Execute	执行 SQL 语句

下面以具体实例介绍如何使用 TDatabase组件。

1. 连接数据库

选中 TDatabase 组件，在对象检查器中设置“AliasName”属性，为 TDatabase 组件选择要连接的数据库，具体操作如图 5.11 所示。

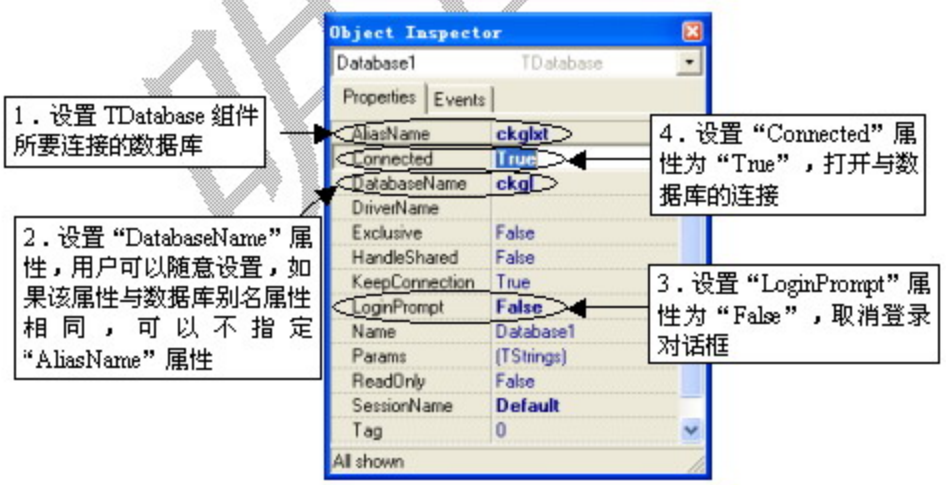


图 5.11　使用 TDatabase 组件连接数据库

2. 使用 TDatabase组件进行事务控制

（1）建立 TDatabase组件与数据库的连接。

（2）设置数据集组件（例如 TQuery）的“DatabaseName”属性为 TDatabase 组件的“DatabaseName”属性值。

（3）编写示例代码。

```
Try
  Database1.StartTransaction;//开始一个事务
  with Query1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('Insert 销售票号表 values (:a,:b,:c,:d)');
    ParamByName('a').AsString := Trim(xsph.Text);
    ParamByName('b').Asinteger := StrToInt(xssl.Text);
    ParamByName('c').AsFloat := StrToFLoat(xsje.Text);
    ParamByName('b').AsString := Trim(czy.Text);
    ExecSQL;
  end;
  with Query1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('Insert 销售表 values (:a,:b,:c,:d)');
    ParamByName('a').AsString := Trim(spbh.Text);
    ParamByName('b').AsString := Trim(spmc.Text);
    ParamByName('c').Asinteger := StrToInt(spsl.Text);
    ParamByName('b').AsFloat := Trim(spje.Text);
    ExecSQL;
  end;
  Database1.Commit; //提交事务
  Application.MessageBox('操作成功.','提示',64);
Except
  Database1.RollBack;//回滚事务
  Application.MessageBox('操作失败.','提示',64);
end;
```

5.2 ADO 数据库程序开发技术

5.2.1 ADO 技术简介

ADO（ActiveX Data Object）是微软推出的新一代数据访问规范，其使用简单，功能强大。ADO 源于微软的另一项技术 OLE DB。OLE DB 是系统级的接口，定义了一套从关系数据库及文件系统的访问数据的 COM（组件对象模型）接口。读者在使用 ADO 开发数据库应用程序时，可以将其视为一个数据库引擎，不必关心它是如何实现的。

5.2.2 TADOConnection 组件

TADOConnection 组件是用来管理 ADO 连接的，并且提供了对事务的支持，类似于

TDatabase 组件。当使用多个 ADO 组件访问数据库时，可以通过 TADOConnection 组件建立一个连接，然后让其他 ADO 组件共享该连接。TADOConnection 组件的重要属性如表 5.4 所示。

表 5.4　　TADOConnection 组件的重要属性、方法

属性、方法	描　　述
ConnectionString	指定连接到 ADO 数据源所需的连接信息
InTransaction	指定事务是否正在处理
Connected	是否打开与数据库的连接
LoginPrompt	确定是否出现登录提示对话框
CloseDatasets	关闭所有的数据集，但不会关闭数据库的连接
BeginTrans	开始一个事务
CommitTrans	提交事务
RollbackTrans	回滚事务
Close	关闭连接

下面介绍如何使多个 ADO 组件共享一个连接。

（1）在窗体中放置 TADOConnection 组件、TADOTable 组件、TDataSource 组件、TDBGrid 组件，如图 5.12 所示。

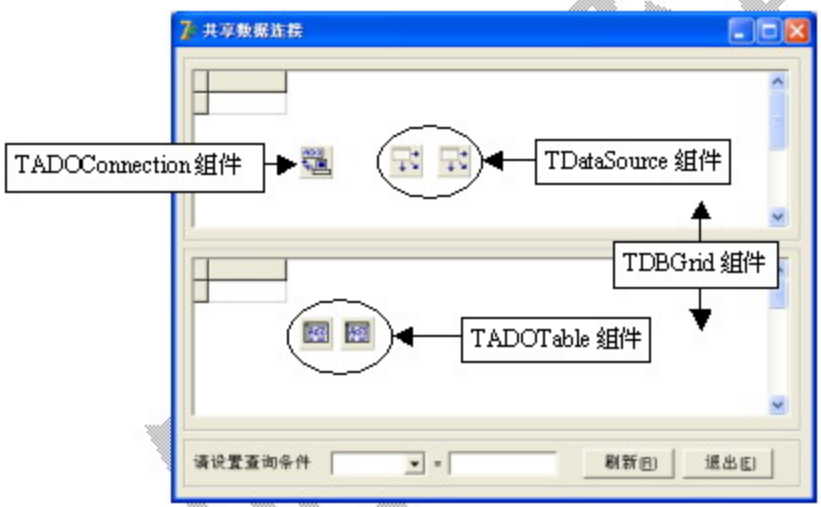
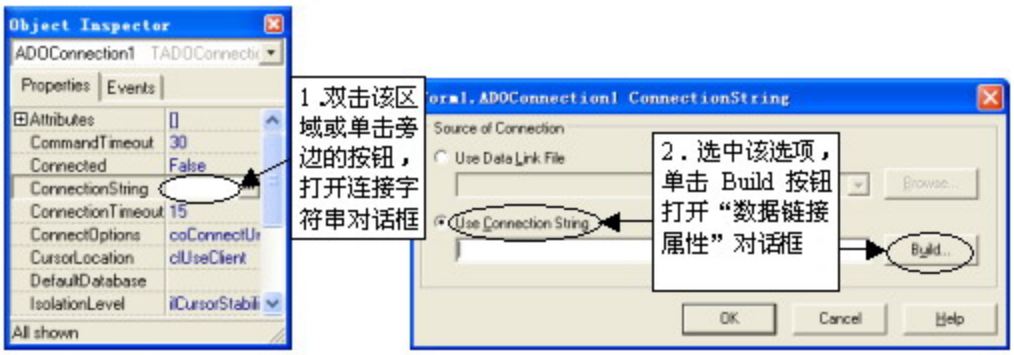


图 5.12　使用 TADOConnection 组件连接数据库

（2）建立 TADOConnection 组件与数据库的连接，具体操作如图 5.13 所示。



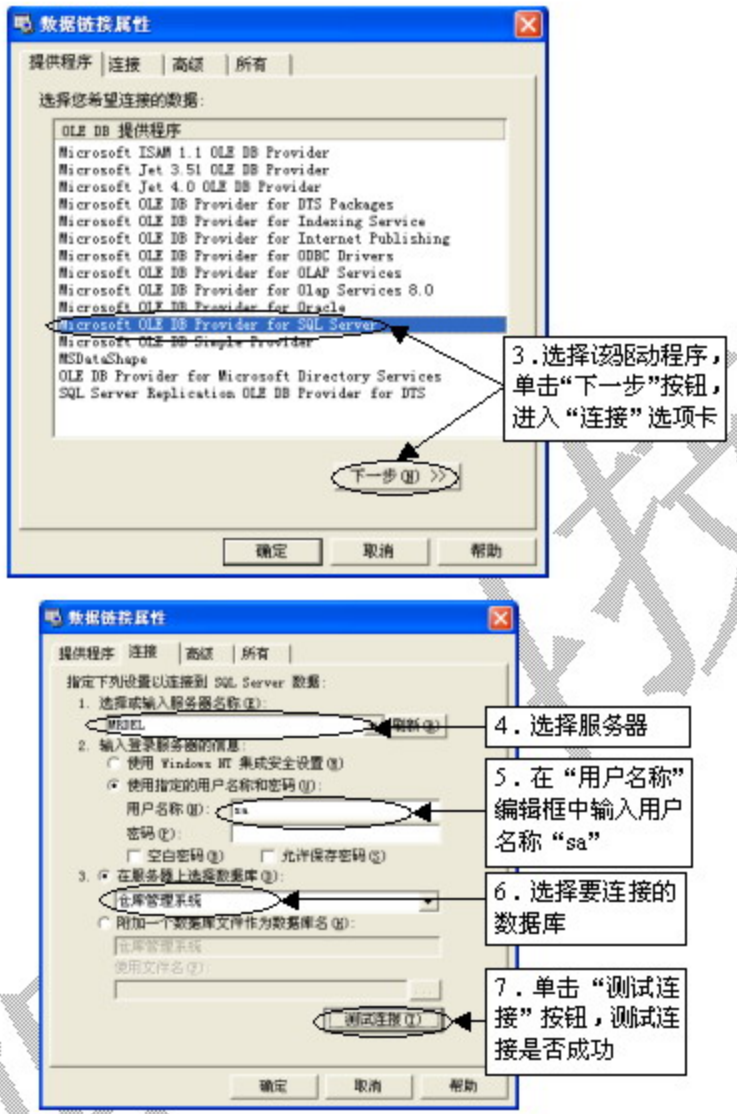


图 5.13　连接数据库

（3）将多个 TADOTable 组件共享一个连接，组件设置如图 5.14 所示。

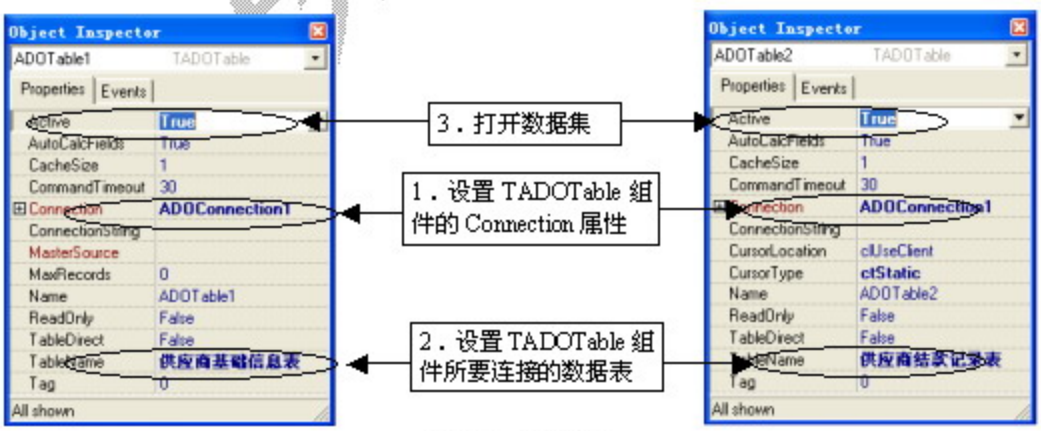


图 5.14　共享连接

（4）显示数据，组件设置如图 5.15 所示。

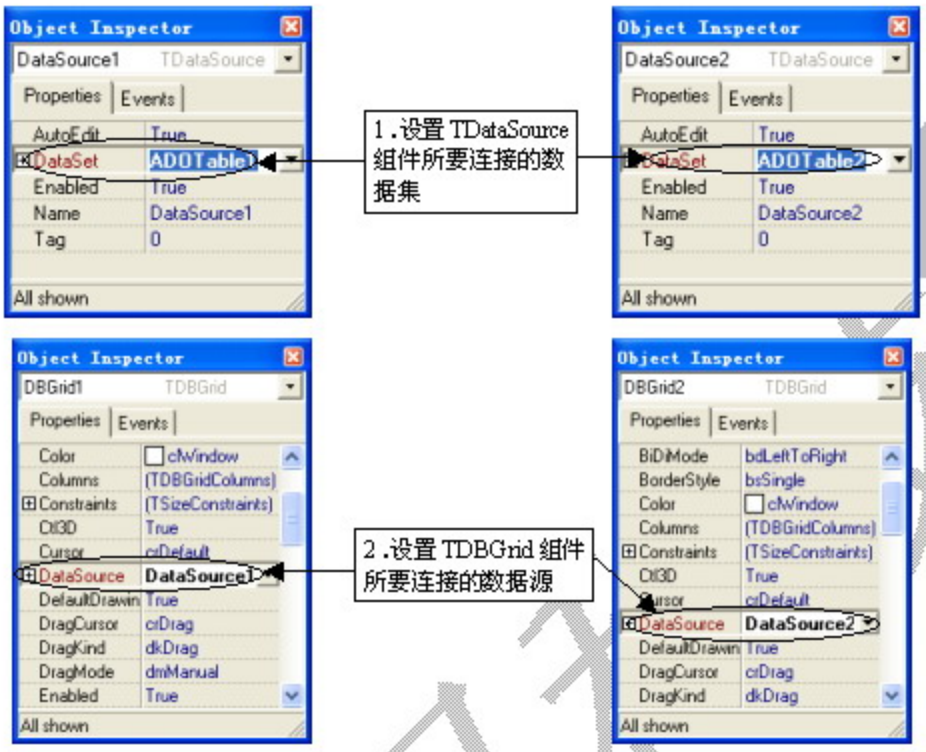


图 5.15　显示数据

（5）运行程序，结果如图 5.16 所示。



图 5.16　共享数据连接

5.2.3　TADOCommand 组件

TADOCommand 组件主要用来执行数据定义的 SQL 语句，即没有返回结果集的 SQL 语句。该组件通过调用 Execute 方法执行 CommandText 属性中的 SQL 语句，其重要属性、方法如表 5.5 所示。

表 5.5　　TADOCommand 组件的重要属性、方法

属性、方法	描　　述
CommandText	设置或读取 TADOCommand 执行的 SQL 指令字符串
Connection	指定通过哪个 TADOConnectiong 组件连接数据库
ConnectionString	数据库的连接信息
Execute	执行 SQL 语句

下面介绍如何使用 TADOCommand 组件操作数据库。

（1）通过 ConnectionString 属性连接数据库，请参考 5.2.2 小节步骤（2）至步骤（4）。

（2）编写程序代码。

```
adocommand1.CommandText := 'insert 操作员信息表 values (:a,:b,:c,:d,:e)'; //添加SQL语句
adocommand1.Parameters.ParamByName('a').Value := 'S001';//为参数赋值
adocommand1.Parameters.ParamByName('b').Value := '王海亮';
adocommand1.Parameters.ParamByName('c').Value := 'WHL';
adocommand1.Parameters.ParamByName('d').Value := 1;
adocommand1.Parameters.ParamByName('e').Value := Now();
adocommand1.Execute;//执行SQL语句
```

5.2.4　TADOTable 组件

TADOTable 是一个数据集组件，可以从 ADO 数据资源中存取数据表。该组件与 TTable 组件类似，可以对数据集进行查找和过滤。其重要属性、方法如表 5.6 所示。

表 5.6　　TADOTable 组件的重要属性、方法

属性、方法	描　　述
Active	确定是否打开数据集
Connection	指定通过哪个 TADOConnection 组件连接数据库
ConnectionString	数据库的连接信息
Append	在数据集末尾添加一个空记录
AppendRecord	将一个新记录添加到给定某些或全部字段值的表末尾
Insert	将数据集置于插入方式，并在当前指针处插入一个空记录
InsertRecord	在当前指针处插入一个给出部分或全部字段值的新记录
TableName	确定 TADOTable 组件所要连接的数据表
Open	打开与数据集的连接
Close	关闭与数据集的连接

下面介绍如何使用 TADOTable 组件访问数据表。

（1）创建一个窗体，如图 5.17 所示。

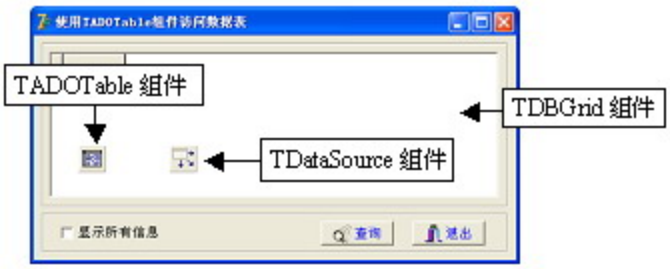


图 5.17　使用 TADOTable 组件访问数据表

（2）通过 ConnectionString 属性建立 TADOTable 组件与数据库的连接，请参考 5.2.2 小节步骤（2）至步骤（4）。

（3）设置各组件的属性，具体操作如图 5.18 所示。

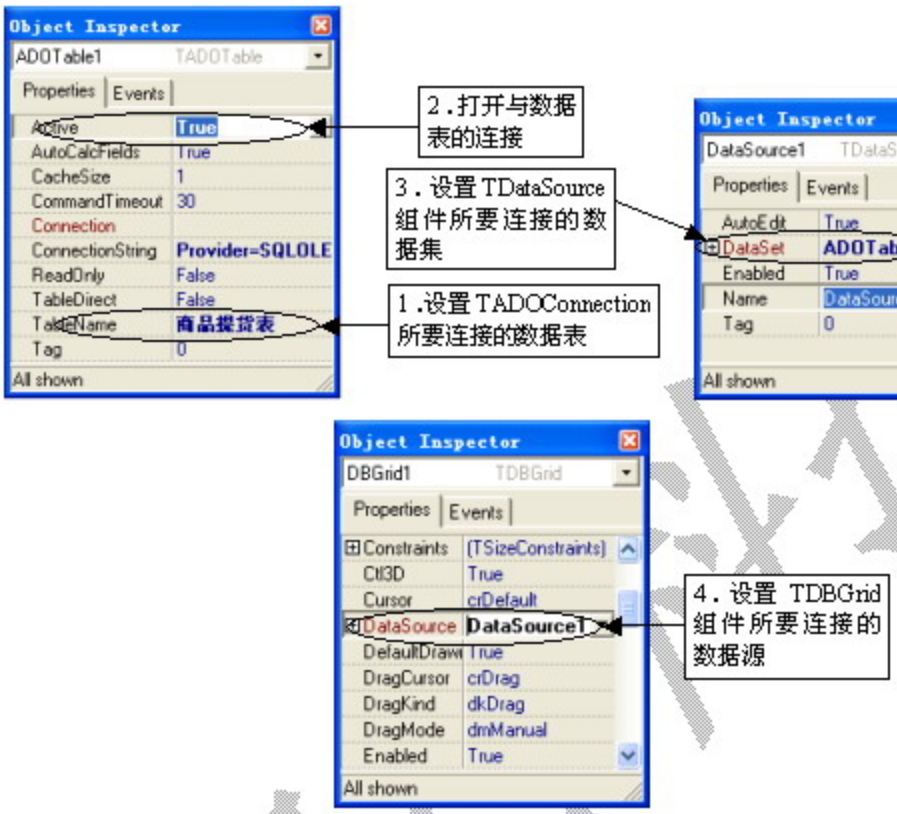


图 5.18　设置属性

（4）运行程序，结果如图 5.19 所示。



图 5.19　使用 TADOTable 组件访问数据表

5.2.5　TADOQuery 组件

TADOQuery 组件提供了在底层的 ADO 数据资源中应用 SQL 语句的功能，使用该组件可以访问多个数据表，其重要属性、方法如表 5.7 所示。

表 5.7　　TADOQuery 组件的重要属性、方法

属性、方法	描　　述
Active	确定是否打开数据集
Connection	指定通过哪个 TADOConnection 组件连接数据库
ConnectionString	数据库的连接信息
RowsAffected	获取最后一次查询操作所处理的数据行数
SQL	指定要执行的 SQL 语句
ExecSQL	执行 SQL 语句，要求无结果集返回
Open	执行 SQL 语句，要求有结果集返回
Close	关闭与数据集的连接

下面介绍如何使用 TADOQuery 组件操作数据库。

（1）创建一个窗体，如图 5.20 所示。

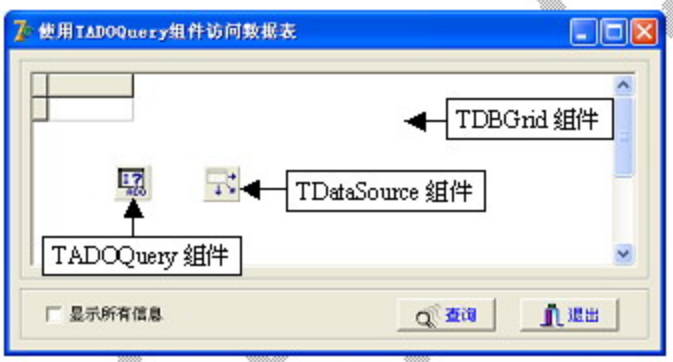
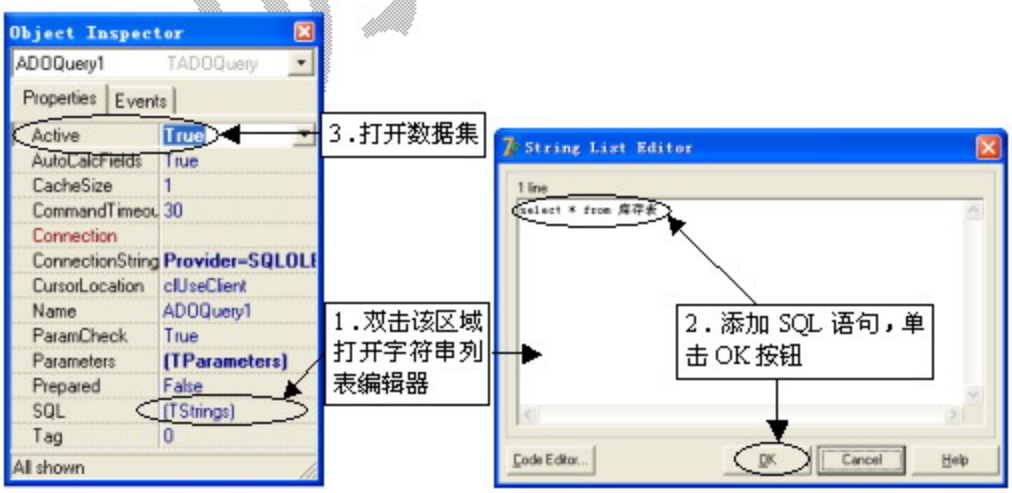


图 5.20　窗体界面

（2）通过设置 ConnectionString 属性建立 TADOQuery 组件与数据库的连接，具体步骤请参考 5.2.2 小节步骤（2）至步骤（4）。

（3）设置各组件的属性，具体操作如图 5.21 所示。



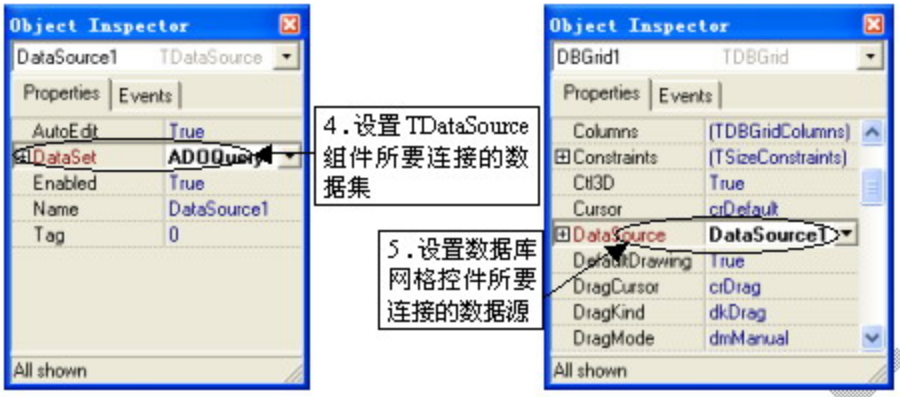


图 5.21　设置组件属性

（4）运行程序，结果如图 5.22 所示。



图 5.22　使用 TADOQuery 组件访问数据表

明日科技

第6章 访问与编辑数据

6.1 操作数据库

6.1.1 创建数据库与数据表

1. 创建数据库

在SQL Server中，可以通过“CREATE DATABASE”语句来创建数据库，其语法格式如下：

```
CREATE DATABASE database_name
ON
( NAME = logical_file_name ,
FILENAME = 'os_file_name' )
```

说明：CREATE DATABASE为关键字，database_name为欲创建的数据库名称，该名称在当前服务器中必须是惟一的。ON关键字指定用来存储数据库数据的磁盘文件（数据文件）。logical_file_name用来在创建数据库后执行的Transact-SQL语句中引用文件的名称。logical_file_name在数据库中必须惟一，并且符合标识符的规则。os_file_name表示数据库文件的物理名称和路径。

图6.1是人事管理系统数据初始化界面，在首次启动人事管理系统时，需要创建数据库，为数据库指定名称、逻辑存储路径、物理存储路径。

在Delphi中，为了执行SQL语句，需要使用相关数据集组件，例如TQuery、TADOQuery等。在使用这些组件时，需要设置其DatabaseName属性建立与数据库的连接，然后通过设置相关属性、调用相应方法提交SQL语句。例如：



图 6.1　创建数据库窗体

```
with ADOQuery1    do
begin
Close;   //关闭数据集
SQL.Clear;//清空SQL语句
SQL.Add();//添加SQL语句
ExecSQL;//执行SQL语句，如果SQL语句有结果集返回，例如执行查询语句，应调用Open方法
end;
```

相关代码如下：

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Try
    with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Create Database '+Trim(Edit1.Text)+' on ( name = '+Trim(Edit2.Text)+' ,
FileName = '+""+Trim(Edit3.Text)+""+' )');
      ExecSQL;
    end;
    Application.MessageBox('数据库创建成功.','提示',64);
  Except
    Application.MessageBox('数据库创建失败.','提示',64);
  end;
end;
```

2. 创建数据表

在 SQL Server 中提供了“CREATE TABLE”语句创建数据表，语法格式如下：

```
CREATE TABLE   table_name
({ column_name data_type
  | column_name AS computed_column_expression
  | < table_constraint > } [,...n]
)
[ ON { filegroup | DEFAULT } ]
```

说明：CREATE TABLE 为关键字，table_name 为欲创建的表名。column_name 是定义

的字段名称，字段名必须符合标识符规则，并且在表内惟一。data_type是字段的数据类型，computed_column_expression是定义计算列值的表达式。计算列是物理上并不存储在表中的虚拟列。计算列由同一表中的其他列通过表达式计算得到。table_constraint 是表的约束，该部分主要用于定义表的主键、外键或表的索引等。ON { filegroup | DEFAULT }指定存储表的文件组。如果指定 filegroup，则表将存储在指定的文件组中。数据库中必须存在该文件组。如果指定 DEFAULT，或者根本未指定 ON 参数，则表存储在默认文件组中。

在创建数据表前，需要指定数据表结构。例如，字段名称、字段类型、是否为主键、是否允许为空等信息。在图 6.2 中，程序以表格形式确定数据表结构，根据用户输入的表结构创建数据表。



图 6.2　创建数据表窗体

程序相关代码如下：

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Try
    with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Create table 操作员信息表 (操作员编号 varchar(20) not NULl,操作员名称
varchar(20),操作员密码 varchar(20),操作员级别 varchar(20))');
      ExecSQL;
    end;
    Application.MessageBox('创建成功.','提示',64);
  Except
    Application.MessageBox('创建失败.','提示',64);
  End;
end;
```

6.1.2　修改数据表结构

在 SQL Server 中修改数据表需要使用“ALTER TABLE”语句，其语法格式如下：

```
ALTER TABLE table_name
{ [ALTER COLUMN column_name
   { new_data_type [ ( precision [ , scale ] ) ]
    [ COLLATE < collation_name > ]
```

```
    [NULL | NOT NULL]
    {ADD | DROP } ROWGUIDCOL
}]
|ADD { column_name data_type }
  |DROP COLUMN    column_name
}
```

说明：ALTER TABLE 是关键字，table_name 是欲修改的表名。ALTER COLUMN 关键字指定要修改的列，column_name 是要修改的列名。new_data_type 是为列定义的新的数据类型。precision 指定数据类型的精度，scale 指定数据类型的小数位数。COLLATE < collation_name > 为更改列指定新的排序规则。排序规则名称既可以是 Windows 排序规则名称，也可以是 SQL 排序规则名称，如果未指定，则此列采用数据库的默认排序规则。NULL | NOT NULL 指定该列是否允许为空。{ADD | DROP } ROWGUIDCOL 指定在指定列上添加或除去 ROWGUIDCOL 属性。ROWGUIDCOL 是一个关键字，表示列是行全局惟一标识符列。ADD 指定要为表添加列，column_name 为新添加的列名，data_type 为新添加列的数据类型。DROP COLUMN 表示要删除一列，column_name 是要删除的列名。

下面的程序代码演示了如何为“操作员信息表”添加新列，同时修改“操作员名称”字段，使其不可以为空，代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  Try
    with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('alter table 操作员信息表 alter column 操作员名称   varchar(20) not null ');
      SQL.Add('alter table 操作员信息表 add 部门 varchar(20)') ;
      ExecSQL;
    end;
    Application.MessageBox('修改成功.','提示',64);
  Except
    Application.MessageBox('修改失败.','提示',64);
  End;
end;
```

6.1.3 删除表

在 SQL Server 中，提供了“DROP TABLE”语句来删除表，其语法格式如下：

```
DROP TABLE table_name
```

说明：DROP TABLE 为关键字，table_name 为欲删除的表名。DROP TABLE 不能用于除去由 FOREIGN KEY 约束引用的表，如果要除去该表，必须先除去引用的 FOREIGN KEY 约束或引用的表。

如果用户删除的数据表不在当前数据库中，在删除数据表时，需要在 SQL 语句中指定表所在的数据库与数据库所有者。例如，下面的代码将删除仓库管理系统中的库存表。

```
DROP TABLE 仓库管理系统.dbo.库存表
```

在图 6.3 中，程序使用组合框显示欲删除的数据表，供用户选择。

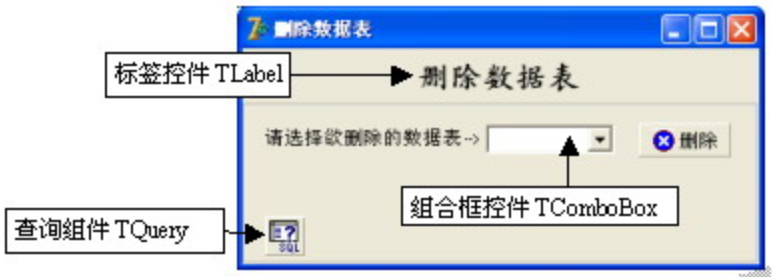


图 6.3　删除数据表窗体

程序相关代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  if Application.MessageBox('确实要删除数据表吗？','提示',mb_yesNo)= id_yes then
  begin
    Try
      with Query1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('Drop Table '+Trim(ComboBox1.Text));
        ExecSQL;
      end;
      Application.MessageBox('数据表删除成功.','提示',64);
    Except
      Application.MessageBox('数据表删除失败.','提示',64);
    End;
  end;
end;
```

注意 初学者经常将 DROP TABLE 语句与 DELERE TABLE、TRUNCATE TABLE 语句混淆，在此，笔者做些介绍。DROP TABLE 语句删除表结构，表的默认值、规则自动解除，与表关联的各种约束、触发器自动除出，而 DELERE TABLE 与 TRUNCATE TABLE 语句只是删除表中的数据，表结构依然存在，DELERE TABLE 语句可以指定删除条件，即可以将符合条件的记录删除，而 TRUNCATE TABLE 语句只能删除表中的所有记录，不能指定删除条件，从执行效率上看，TRUNCATE TABLE 语句要快，且使用的系统日志资源和事务日志资源较少。

6.2　编辑数据

6.2.1　添加记录

数据库管理系统中最常见的操作就是添加数据，在 Delphi 中，可以使用 TTable、

TADOTable、TQuery、TADOQuery等数据集组件添加数据，因为TTable与TADOTable组件、TQuery与TADOQuery组件方法基本相同，下面主要介绍TADOTable、TADOQuery组件添加数据的方法。

1. 利用TADOTable组件添加记录

TADOTable组件除了提供连接数据表的功能，还实现了对数据表的各种操作，例如插入、删除、查找数据。用户通过该组件可以不使用任何SQL语句便可实现各种数据库操作。有关TADOTable组件的常用属性、方法请参考5.2.4小节的TADOTable组件。

TADOTable组件提供了Append、AppendRecord、Insert、InsertRecord共4个方法来添加数据。Append在数据集的末尾添加一条空记录，AppendRecord在数据集的末尾插入数据，Insert在数据集的当前行插入一条空记录，InsertRecord在数据集的当前行插入数据。

Append方法原型如下：

```
procedure Append;
```

说明：该方法在数据集的末尾打开一个新的空记录，它只对可修改的结果集起作用，否则会产生异常。

AppendRecord方法原型如下：

```
procedure AppendRecord(const Values: array of const);
```

说明：该方法将一个新记录追加到数据表的末尾，并根据参数Values设置部分或全部的字段值。

Values是能被转换成数据表相应字段的数据类型的常量值数组。数组中元素的数据类型要与表字段类型相同或兼容。

Insert方法原型如下：

```
procedure Insert;
```

说明：该方法在数据集当前光标位置打开一个新的空记录，在调用Insert方法时，数据集必须打开，并且处于可编辑状态。

InsertRecord方法原型如下：

```
InsertRecord(const Values: array of const);
```

说明：该方法在当前记录处插入一个给出部分或全部字段值的新记录。

Values是一个常数值数组，能够将其转换为数据表中相应字段的数据类型。

由于Append与Insert方法、AppendRecord与InsertRecord方法在使用上基本相同，在此只介绍Append与AppendRecord添加数据的方法。

❑ 使用Append方法添加数据

（1）调用Open方法打开数据集。

（2）调用Edit方法使数据集处于编辑状态。

（3）调用Append方法在数据集末尾添加空记录。

（4）为空记录的各个字段赋值。

（5）调用Post方法提交数据。

如：

```
with ADOTable1 do
```

```
begin
  Open;
  Edit;
  Append;
  FieldByName('商品编号').AsString := Trim(Edit1.Text);
  FieldByName('商品名称').AsString := Trim(Edit2.Text);
  FieldByName('规格').AsString := Trim(Edit3.Text);
  FieldByName('产地').AsString := Trim(Edit4.Text);
  FieldByName('数量').AsInteger := StrToInt(Edit5.Text);
  FieldByName('金额').AsFloat := StrToFloat(Edit6.Text);
  Post;
end;
```

- 使用 AppendRecord 方法添加数据

（1）调用 Open 方法打开数据集。

（2）调用 AppendRecord 方法添加数据。

如：

```
ADOTable1.Open;
ADOTable1.AppendRecord([Trim(ygbh.Text),Trim(ygxm.Text),Trim(ssbm.Text),
Trim(xb.Text),StrToInt(nl.Text),StrToFloat(gz.Text)]);
```

2. 利用 TADOQuery 组件添加记录

TADOQuery 组件提供了使用 SQL 语句操作数据库的功能。结构化查询语言 SQL 包括数据定义语言、数据操纵语言，能够实现各种复杂的数据库操作。有关 SQL 语句的介绍请参考第 7 章的 SQL 语句。

在使用 TADOQuery 组件执行 SQL 语句时，经常需要在 SQL 语句中指定参数，以进行更灵活的操作，步骤如下。

（1）关闭数据集。

（2）清空 SQL 语句。

（3）添加 SQL 语句。

（4）为 SQL 语句中的参数赋值。

（5）调用 Open 或 ExecSQL 方法执行 SQL 语句。如果 SQL 语句要求有结果集返回，应调用 Open 方法，例如，执行 Select 语句；如果 SQL 语句不要求有结果集返回，如执行添加、修改、删除等操作，应调用 ExecSQL 方法执行 SQL 语句。

如：

```
with DataADOQuery1 do
begin
Close;//关闭数据集
SQL.Clear;//清空SQL语句
SQL.Add('select * from 库存上下限表 where 商品编号 = :a ');//添加SQL语句
Parameters.ParamByName('a').Value := Trim(ComboBox2.Text);//为参数a赋值
Open;//执行SQL语句
end;
```

在 Delphi 中，向 SQL 语句中添加参数，需要在参数前加“:”，然后在执行 SQL 语句

前为参数赋值。

在添加数据时，有些字段可能不允许为空，虽然在设计数据表时通过建立各种约束防止了空数据的插入，但是在程序设计时，笔者建议对用户录入的数据进行判断，如果必填项数据为空，则进行提示，并停止添加操作。这样做，可以防止数据因违背数据表约束导致添加失败。例如，在商品报损管理模块中，如果用户在表格中录入的商品信息有空项，程序将停止保存操作，同时显示提示信息，如图 6.4 所示。



图 6.4　商品报损处理

程序相关代码如下：

```
procedure TForm18.BitBtn4Click(Sender: TObject);
var
  i: Integer;
begin
  if JCxmlb = True then   //自定义函数，用于判断表格中是否有空项
  begin
    Try
      For i := 1 to StringGrid1.RowCount-1 do
      begin
        with Data.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('insert 报损表 values (:a,:b,:c,:d,:e,:f,Default)');
          Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,StringGrid1.RowCount-1]);
          Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[1,StringGrid1.RowCount-1]);
          Parameters.ParamByName('c').Value := StrToInt(StringGrid1.Cells[2,StringGrid1.RowCount-1]);
          Parameters.ParamByName('d').Value := StrToFloat(StringGrid1.Cells[3,StringGrid1.RowCount-1]);
          Parameters.ParamByName('e').Value := Trim(Label8.Caption);
          Parameters.ParamByName('f').Value := Trim(StringGrid1.Cells[5,StringGrid1.RowCount-1]);
          ExecSQL;
        end;
      end;
```

```
        Application.MessageBox('报损处理成功。','提示',0+64);
      except
        Application.MessageBox('系统出错。','提示',0+64);
      end;
    end
    else
      Application.MessageBox('项目列表有误。','提示',64);
end;
```

注意：在向数据表中插入数据时，数据类型要与表格字段类型相同或兼容，否则需要进行类型转换。例如，在本节实例中，表格中的数据均为字符串类型，而数量字段为整型，时间字段为日期型，金额字段为货币型，在保存数据时都需要进行类型转换。

6.2.2 修改记录

1. 利用 TADOTable 组件修改数据

在 Delphi 中实现数据的修改可以采用 3 种方式，一是使用数据库表格组件，调用数据库表格组件的相关方法实现数据的修改；二是使用 SQL 语句中的“Update”语句来实现数据的修改；三是利用数据感知控件实现数据的修改，将数据感知控件绑定到数据集上，通过修改数据感知控件中的数据来实现修改。本节采用第一种方式，利用 TADOTable 组件实现数据的修改。

使用 TADOTable 组件修改数据也有多种方法。例如，可以为数据集的各个字段赋值，然后调用 Post 方法提交数据，也可以直接调用 SetFields 方法修改数据，但无论哪种方法，只有在数据集打开并处于插入或编辑状态时才可以修改数据。

SetFields 方法原型如下：

```
procedure SetFields(const Values: array of const);
```

说明：该方法在当前记录上设置部分或全部字段值。

Values 是一个常量值的数组，可以转换为数据集中相应字段的数据类型。

- 通过为字段赋值修改数据

（1）调用 Edit 方法使数据集处于编辑状态。

（2）为各个字段赋值。

（3）调用 Post 方法保存数据。

如：

```
with ADOTable1 do
begin
  Edit;
  FieldByName('操作员编号').AsString := Trim(Edit1.Text);
  FieldByName('操作员名称').AsString := Trim(Edit2.Text);
  FieldByName('操作员密码').AsString := Trim(Edit3.Text);
  FieldByName('操作员权限').AsString := Trim(Edit4.Text);
  Post;
end;
```

❑ 利用 SetFields 方法修改数据

（1）调用 Edit 方法使数据集处于编辑状态。

（2）调用 SetFields 方法保存数据。

如：

```
ADOTable1.Edit;
ADOTable1.SetFields([Trim(khbh.Text),Trim(khmc.Text),Trim(jc.Text),
Trim(dz.Text),Trim(lxr.Text),Trim(lxdh.Text)]);
```

2. 利用 TQuery 组件修改数据

使用 TADOTable 组件或 TTable 组件只能进行简单的数据修改，但在实际应用中，往往需要进行复杂的数据修改。在 13.6.1 小节的数据修改中，作者介绍了“Update”语句，使用该语句便可以实现数据的批量修改，如果想只修改符合条件的记录，可以在“Update”语句中指定“Where”条件。

使用 TQuery 组件执行 SQL 语句与 TADOQuery 组件略有不同，主要区别在于为参数赋值，下面是使用 TQuery 组件执行 SQL 语句的程序代码。

```
With Query1 do
begin
Close;//关闭数据集
SQL.Clear;//清空SQL语句
//添加SQL语句
SQL.Add('Update 员工信息表 set员工姓名 =:b,员工工资 = :c where 员工编号 = :a');
ParamByName('a').AsString := Trim(Edit1.Text);//为参数赋值，指定参数类型
ParamByName('b').AsString := Trim(Edit2.Text);
ParamByName('c').AsFloat := StrToFloat(Edit3.Text);
ExecSQL;//执行SQL语句
end;
```

图 6.5 是商品库存价格调整模块，在该模块中需要修改指定商品的进货价格，为了满足实际需求，在修改商品进货价格时，可以按原商品的百分比进行调价，也可以按实际价格调价。



图 6.5 库存价格调整

程序相关代码如下：

```
with Data.ADOQuery1 do
```

```
begin
Close;
SQL.Clear;
SQL.Add('update 库存表 set 库存金额 = :a where 商品编号 = :b');
Parameters.ParamByName('a').Value := jg*Sum;
Parameters.ParamByName('b').Value := Trim(Data.ADOQuery2.FieldByName('商品编号
').Value);
ExecSQL;
end;
```


6.2.3 删除记录

1. 利用 TADOTable 组件删除数据

使用数据库表格组件除了可以添加、修改数据，还可以删除数据。在 TADOTable 组件中提供了 Delete 方法用于删除记录，但是该方法只对一个可修改的结果集有影响，并且在删除数据时，数据集应处于打开状态，否则会产生异常。此外，在删除数据时，注意保证数据的完整性。

Delete 方法原型如下：

```
procedure Delete;
```

说明：该方法删除当前记录，将数据集指针移到下一条记录上。

删除数据是一项危险的操作，如果删除不当，会给用户带来损失。在开发程序过程中，通常的做法是在删除数据前进行提示，让用户确认是否删除数据，防止因误删数据带来损失。

使用 Delete 方法删除数据的基本步骤如下。

（1）判断数据集是否激活。

（2）提示用户是否删除数据。

（3）调用 Delete 方法删除数据。

（4）提示操作成功。

如：

```
if Application.MessageBox('确实要删除当前数据吗?','提示',mb_YesNo)= ID_Yes then
begin
  if ADOTable1.Active then   //判断数据集是否打开
  begin
    Try
      ADOTable1.Delete;//删除当前记录
      Application.MessageBox('删除成功.','提示',64);
    Except
      Application.MessageBox('系统出错.','提示',64);
    end;
  end
  else
    Application.MessageBox('数据集没有打开.','提示',64);
end;
```

2. 利用 TADOQuery 组件删除数据

使用 TADOTable 组件的 Delete 方法一次只能删除一条记录，并且是当前记录。如果要删除多条记录或删除指定条件的记录，应使用 Delete 语句来实现，有关 Delete 语句的介绍请参考 13.7 小节的数据删除。

使用 TADOQuery 组件删除数据的基本步骤如下。

（1）提示是否删除数据。

（2）关闭数据集。

（3）清空 SQL 语句。

（4）添加 Delete 语句。

（5）调用 ExecSQL 方法执行 SQL 语句。

如：

```
if Application.MessageBox('确实要删除当前记录吗？','提示',MB_YESNO )= ID_Yes then
begin
  with ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('delete 部门信息表 where 部门编号 = :a');
    Parameters.ParamByName('a').Value := Trim(Label1.Caption);
    ExecSQL;
  end;
end;
```

图 6.6 描述的是销售管理系统中商品基础信息管理模块，在该模块中采用 Delete 语句来实现商品信息的删除，在删除数据时，通过在 Delete 语句中指定 Where 子句来删除符合条件的数据。

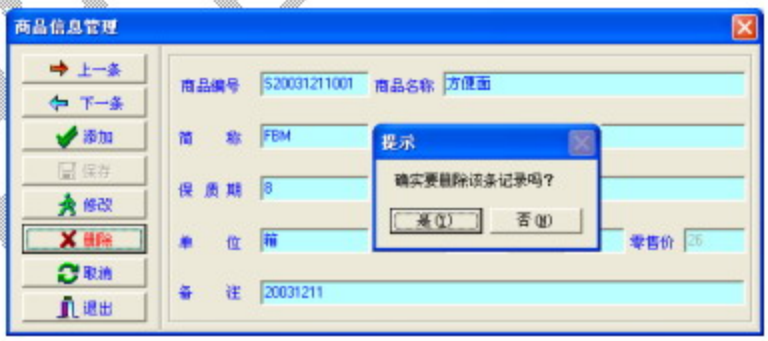


图 6.6 商品信息管理

程序相关代码如下：

```
procedure TForm7.BitBtn6Click(Sender: TObject);
begin
  BitBtn4.Enabled := False;
  Try
  if Application.MessageBox('确实要删除该条记录吗？','提示',MB_YESNO )= ID_Yes then
  begin
```

```
        with Data.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('delete 商品基础信息表 where 商品编号 = :a');
          Parameters.ParamByName('a').Value := Trim(Edit1.Text);
          ExecSQL;
        end;
        Application.MessageBox('该条记录已经删除。','提示',0+64);
        OnShow(Sender);
      end;
      Except
        Application.MessageBox('系统出错。','提示',0+64);
        Close;
      end;
    end;
```


6.3 数据查询与统计

6.3.1 数据查找

1. Goto 型查找

使用数据集组件 TTable 可以方便地浏览数据，同时，该组件还提供了多种方法用于查询数据，例如，提供的 Goto 型查找有 GotoNearest、GoKey 等，Find 型查找有 FindNearest、FindKey 等。使用这些方法，用户可以不用任何 Select 语句查询数据。

GotoNearest 方法会将当前指针移动到与当前键值最匹配的记录上。原型如下：

```
procedure GotoNearest;
```

说明：调用该方法会改变当前指针，使其移动到与键集缓冲区中当前键值匹配的记录上，如果没有匹配的记录，当前指针会停留在数据表的末尾。在调用该方法前，应用程序必须调用 SetKey 或 EditKey 方法使数据集处于“dsSetKey”状态。

GotoKey 方法移动指针到当前键值标识的记录上。原型如下：

```
function GotoKey: Boolean;
```

说明：该方法查找与键值匹配的记录，如果找到，函数返回值为 True，当前指针会移到与键值匹配的记录上；如果没有找到数据，返回值为 False，当前指针不改变。

使用 GotoNearest 方法查找数据的步骤如下。

（1）设置索引字段。

（2）调用 SetKey 或 EditKey 方法使数据集处于“dsSetKey”状态。

（3）用查找值填充键集缓冲区。

（4）调用 GotoNearest 方法查找数据。

如：

```
Table1.IndexFieldNames := '药品编号';
```

```
Table1.SetKey;
Table1.FieldByName('药品编号').AsString := 'YP002';
Table1.GotoNearest;
```

使用 GotoKey 查找数据与 GotoNearest 基本相同，只是步骤（4）中调用 GotoKey 方法，在此不做过多介绍。

图 6.7、图 6.8 是商品基础信息查询设计期窗体和运行期窗体，程序中使用 GotoNearest 方法查找数据。当用户设置查询条件后，程序会搜索与查询条件匹配的记录。



图 6.7　商品信息查询设计期窗体　　图 6.8　商品信息查询运行期窗体

程序相关代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  if Trim(ComboBox1.Text)<>'' then
  begin
    Table1.IndexFieldNames := Trim(ComboBox1.Text); //设置索引字段
    Table1.SetKey;//设置键值
    Table1.FieldByName(Trim(ComboBox1.Text)).AsString := Trim(Edit1.Text);
    Table1.GotoNearest;//查找匹配的数据
  end;
end;
```

2. Find 型查找

TTable 组件除了可以进行 Goto 型数据查找，还可以进行 Find 型数据查找，提供相应的方法有 FindKey 和 FindNearest。其中 FindKey 方法会查找包含指定键值的记录，而 FindNearest 会查找与指定键值匹配的记录。

FindKey 方法原型如下：

```
function FindKey(const KeyValues: array of const): Boolean;
```

说明：该方法在数据集中查找标识的记录。键集中的每个键值可以是文字、变量或 Nil。如果键值的数量小于索引列的数量，则缺少的键值被认为是 Null，如果键值的数量多于索引列的数量，会出现字段索引越界的错误。FindKey 成功，返回值为 True，当前指针会停留在找到的记录上；否则，返回值为 False，当前指针不改变。

KeyValues 是一个包含逗号分割的由字段值组成的常量数组。

FindNearest 方法原型如下：

```
procedure FindNearest(const KeyValues: array of const);
```

说明：该方法查找与指定键值匹配的记录，并将指针定位到找到的记录。参数 KeyValues 与 FindKey 方法中的参数相同。

在使用 FindKey 方法或 FindNearest 方法查找数据前，需要设置索引字段，可以通过设置 TTable 组件的 IndexFieldNames 属性实现。如果要设置多个索引字段，索引字段间用分号分隔，如：

```
Table1.IndexFieldNames := '客户全称'+';'+'地址';
```

由于 FindKey 方法与 FindNearest 方法在使用上基本相同，在此只介绍使用 FindKey 方法查找数据的一般步骤。

（1）调用 Open 方法打开数据集。

（2）设置索引字段。

（3）调用 FindKey 方法查找数据。

如：

```
if Trim(ComboBox1.Text)<>'' then
begin
  Table1.Open;
  Table1.IndexFieldNames := '药品编号'; //设置索引字段
  Table1.FindNearest(['YP002']);
end;
```

3. Locate 型查找

使用 TTable 组件的 Locate 方法可以查找与索引字段数据相同或匹配的记录，该方法原型如下：

```
function Locate(const KeyFields: String; const KeyValues: Variant; Options: TLocateOptions): Boolean;
```

说明：如果函数执行成功，返回值为 True，查询到的记录会成为当前记录，反之，返回值为 False，当前记录不改变。

KeyFields 标识索引字段。

KeyValues 标识索引键值。

Options 确定搜索方式，它是一个集合类型，有两个元素，分别为“loCaseInsensitive”和“loPartialKey”。“loCaseInsensitive”表示搜索时不区分字母大小写，“loPartialKey”表示参数 KeyValues 的值为查找字段数据的一部分。

使用 Locate 方法查找数据的步骤如下。

（1）打开数据集。

（2）设置索引字段。

（3）调用 Locate 方法查找数据。

如：

```
Table1.Open;
Table1.IndexFieldNames := '供应商全称'; //设置索引字段
Table1.Locate('供应商全称','哈尔滨食品厂',[loCaseInsensitive,loPartialKey]) ;
```

图 6.9、图 6.10 是销售管理系统中客户结款信息查询设计期窗体与运行期窗体，在查询客户结款信息时，程序使用 Locate 方法定位记录。当用户设置查询条件后，程序在表格中自动定位到符合条件的记录上。

程序相关代码如下：



图 6.9　客户结款信息查询设计期窗体



图 6.10　客户结款信息查询运行期窗体

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
   if Trim(ComboBox1.Text)<>'' then
begin
      Table1.Open;
      Table1.IndexFieldNames := Trim(ComboBox1.Text); //设置索引字段
      Table1.Locate(Trim(ComboBox1.Text),Trim(Edit1.Text),[loCaseInsensitive,
loPartialKey]) ;
end;
end;
```

6.3.2　数据查询

在开发进销存管理软件时，数据查询是必不可少的部分。例如，在数据统计、报表打印中，都需要进行数据查询。在 Delphi 中，进行数据查询可以有多种方法，但从其思路上看，主要采用两种方式，一是通过 Delphi 的数据集组件提供的各种方法进行查询，二是通过 SQL 语句中的 Select 语句进行查询，在 6.3.1 节的数据查找中笔者已经介绍了直接使用数据集提供的方法进行数据查找，下面将介绍第二种方式，利用 Select 语句进行数据查询。

使用 Select 语句可以进行各种复杂的查询，而不仅仅是简单地定位记录。在应用 Select 语句时，应根据需求设置查询条件，如果不指定查询条件，语句将返回数据表所有记录。在仓库管理系统的商品出库查询模块中，为了满足用户的需求，程序需要按客户信息、商品信息、提货票号进行查询，同时还要求可以进行时间段查询。要实现这种动态查询，需要在程序中对查询条件进行判断，例如，用户是否选择了按时间段查询，如果不要求按时间段查询，执行相关 SQL 语句，如果按时间段查询，则执行另外的 SQL 语句，如图 6.11 所示。



图 6.11　商品出库查询

程序相关代码如下：

```
procedure TForm37.SpeedButton1Click(Sender: TObject);
begin
  if (Trim(ComboBox2.Text)<>'')and(Trim(ComboBox1.Text)<>'') then
  begin
    CheckBox1.Checked := False;
    with Data.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      if CheckBox2.Checked = False then
      begin
        SQL.Add('select * from 商品提货表 where ');
        SQL.Add(Trim(ComboBox1.Text));
        SQL.Add('=');
        SQL.Add(':a');
      end
      else
      begin
        SQL.Add('Select * from 商品提货表 where');
        SQL.Add(Trim(ComboBox1.Text));
        SQL.Add('=');
        SQL.Add(':a');
        SQL.Add('and 提货票号 in (Select 提货票号 from 商品提货票号表 where 时间>=
                :b and 时间<= :c)');
        Parameters.ParamByName('b').Value := Int(DateTimePicker1.Date)-2 ;
        Parameters.ParamByName('c').Value := Int(DateTimePicker2.Date)-2+ 0.9998;
      end;
      Parameters.ParamByName('a').value := Trim(ComboBox2.Text);
      Open;
    end;
    if Data.ADOQuery2.RecordCount >0 then
      DataSource1.DataSet := Data.ADOQuery2;
  end;
end;
```

6.3.3 数据排序

在开发数据库程序时，有时需要进行数据排序。例如，学生成绩排行、商品销售排行等。在 13.2.4 小节对查询结果进行排序中，笔者介绍了在 Select 语句中如何通过 Order By 子句进行排序。在实际应用中，有时只需提取部分排行结果，例如，查看本月销售额位于前 5 位的商品、查看本月最畅销的 10 种图书等。此时，需要在 Select 语句中进一步指定提取记录数，可以通过 Top 关键字实现，例如：

```
select top 5 * from 销售表  orderby 数量 Desc
```

图 6.12 是商品销售排行模块窗体，当用户指定查询时间后，程序会对商品销售数量进行排序，显示销售数据居于前 3 位或后 3 位的商品信息。



图 6.12　商品销售排行

程序相关代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
var
   Year,month,day: Word;
begin
   DecodeDate(DateTimePicker1.Date,year,month,day);
   with ADOQuery1 do
   begin
      Close;
      SQL.Clear;
      SQL.Add('select top 3 * from 销售表 where 销售票号 in (Select 销售票号 from 销售票号
               表 where year(销售时间) = :a and month(销售时间)= :b ) order by 数量 ');
      if sx.Checked then
         SQL.Add('ASC')
      else
         SQL.Add('Desc');
      Parameters.ParamByName('a').Value := year;
      Parameters.ParamByName('b').Value := month;
      Open;
   end;
   if ADOQuery1.RecordCount<1 then
      ADOQuery1.Close;
end;
```

6.3.4　数据过滤

在开发程序时，有时需要使用数据集组件的一些方法进行数据过滤。读者可能会产生疑问，使用 Select 语句可以实现各种复杂查询，在进行数据过滤时，完全可以使用 Select 语句来实现。是的，但是使用 Select 语句只能对表、视图等数据库对象进行查询，对于一个结果集，如果结果集不是表或视图，则不能使用 Select 语句进行查询，例如，对使用 Restore HeaderOnly 命令返回的结果集进行查询，则不能使用 Select 语句，只能通过数据集组件的一些方法进行过滤。

数据集组件提供了 Filter 属性与 Filtered 属性进行数据过滤，Filter 属性用于指定当前过滤器文本，Filtered 属性确定是否进行数据集过滤。

Filter 属性原型如下：

```
property Filter: String;
```

说明：使用 Filter 属性可以指定记录集的过滤条件，通过设置 Filtered 属性为 True，可以对记录集进行过滤，对于应用程序来说，只有符合过滤条件的记录可以应用。

Filtered 属性原型如下：

```
property Filtered: Boolean;
```

说明：该属性确定是否激活过滤器。当过滤器激活时，如果用户修改当前记录，即使不再符合过滤条件，保存数据时，该记录将不可见，下一条符合条件的记录将成为当前记录。

使用 Filter 属性与 Filtered 属性过滤数据的一般步骤如下。

（1）关闭过滤器。

（2）设置过滤条件。

（3）打开过滤器。

如：

```
ADOTable1.Filtered := False;
ADOTable1.Filter := '商品名称 = "方便面"';
ADOTable1.Filtered := True;
```

6.3.5 数据统计

在 SQL Server 2000 中，提供了许多聚合函数进行数据统计，同时，在 Select 语句中还提供了 Group By 子句进行分组统计。有关聚合函数的用法以及 Group By 子句的语法请参考 13.2.5 小节的将查询结果分组统计与 13.3 小节的使用聚合函数进行查询。

1. 统计总数量与平均值

使用聚合函数可以对一组值进行计算并返回单一值，除 Count 函数外，所有聚合函数忽略空值（Null）。在进行数据分析统计时，经常需要查看汇总数据与平均值。例如，在工资管理系统中统计职工工资数据，用户需要查看某个月份的工资发放金额及人均工资。在 SQL Server 2000 中直接提供了 Sum 与 AVG 聚合函数来显示汇总数据和平均值，省去了手工计算的麻烦。

> 注意：在 Select 语句中使用聚合函数，无论是否有符合条件的记录，语句都返回至少一条记录，如果数据表中没有符合条件的记录，语句返回的数据为空。所以，在程序中判断含有聚合函数的 Select 语句是否有数据返回，不能通过 RecordCount 属性判断，而应判断字段数据是否为空，如：

```
if ADOQuery1.Fields[0].Value = Null then
    ADOQuery1.Close;
```

图 6.13 是工资发放统计分析窗体，当用户设置查询日期后，程序会统计当月的工资发放数据。

程序相关代码如下：

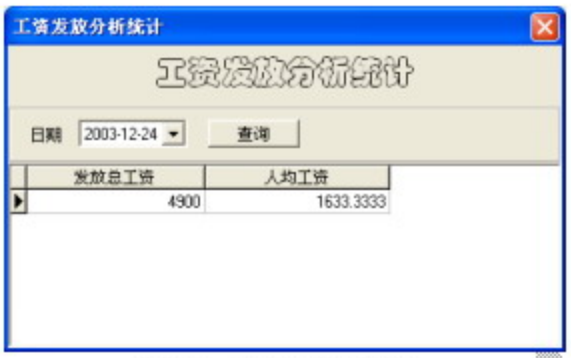


图 6.13　工资发放分析统计

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
var
   year,month,day: Word;
begin
   DecodeDate(DateTimePicker1.Date,year,month,day);
   with ADOQuery1 do
   begin
      Close;
      SQL.Clear;
      SQL.Add('select sum(实发金额)as 发放总工资,avg(实发金额)as 人均工资 from 工资发放明细
               表 where Year(时间)= :a and month(时间) = :b');
      Parameters.ParamByName('a').Value := year;
      Parameters.ParamByName('b').Value := month;
      Open;
   end;
   if ADOQuery1.Fields[0].Value = Null then
      ADOQuery1.Close;
end;
```

2. 分组统计

在 Select 语句中使用聚合函数，只能进行简单的数据统计，就像报表底部的总计。在实际应用中，例如，在商品销售报表中，往往需要显示小计信息，汇总每一种商品，为每种商品产生一条汇总记录。

在 SQL Server 2000 中，提供了 Group By 语句用于进行数据分组。Group By 语句往往与聚合函数组合使用，用于对分组字段数据进行汇总，如：

```
select 商品名称,sum(数量) 数量,sum(金额)销售金额 from 销售表 group by 商品名称
```

在 Select 语句中使用 Group By 有一定限制：

- Group By 子句列出的分组列必须来自于 From 子句列出的表；
- 在 Select 语句中列出的选择列如果不包含字段函数，则必须出现在 Group By 子句中，即必须是分组字段。

使用 Group By 语句时，分组字段可以指定多个字段，即对多个字段进行分组。例如，统计库存表中每个库的每种药品库存信息，程序需要对“库存名称”、“商品名称”字段进

行分组。如：

```
select 库存名称,药品名称,sum(库存数量)库存数量,sum(库存金额)库存金额 from 库存 group by 库存名称,药品名称
```

图 6.14 是商品出库统计查询窗体，程序会按客户名称、商品名称统计商品出库情况。



图 6.14　商品出库统计查询

程序相关代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  Year,Month,Day: Word;
begin
  DecodeDate(DateTimePicker1.Date,Year,Month,Day);
  with ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select 客户名称,商品名称, sum(数量) 数量,sum(金额) 金额 from 商品提货表 where year(时间)=:a and month(时间)= :b Group By 客户名称,商品名称');
    Parameters.ParamByName('a').Value := Year;
    Parameters.ParamByName('b').Value := Month;
    Open;
  end;
end;
```

第 7 章 数据库用户界面设计

7.1 数据控制组件介绍

数据控制组件又称数据感知组件，用来显示数据表中的数据。在 Delphi 的“Data Controls”面板中提供了许多数据控制组件供开发人员使用，利用数据控制组件，甚至不需要写任何 SQL 语句便可以实现对数据库的各种操作。Delphi 提供的数据控制组件如图 7.1 所示。

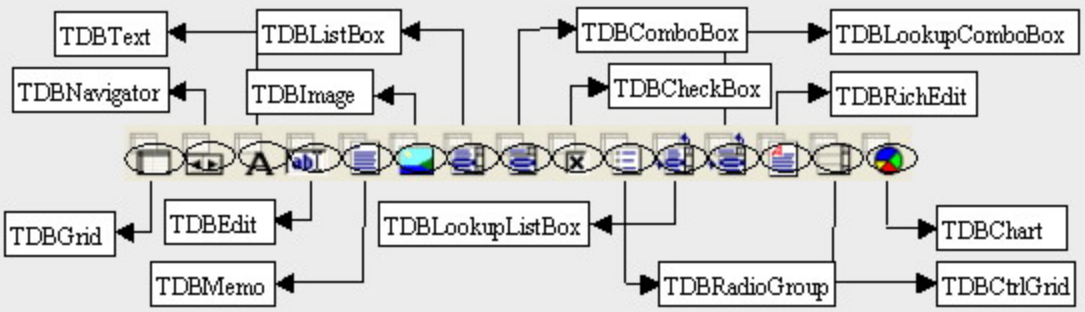


图 7.1 数据控制组件

7.2 TDBEdit 组件介绍

TDBEdit 组件用来显示数据集中某一字段的当前值，如果数据集允许，通过该组件可以修改相应的字段值。其主要属性、方法、事件如下。

- DataSource

该属性确定 TDBEdit 组件所要连接的数据源。

- DataField

确定 TDBEdit 组件所要连接的字段，在设置该属性前，需要先设

置 DataSource 属性。

- Clear

清空 TDBEdit 组件的内容。

- CharCase

该属性确定 TDBEdit 组件文本的大小写，共有 3 个属性值：

- ecLowerCase

TDBEdit 组件文本为小写状态；

- ecNormal

TDBEdit 组件文本为正常状态，它不强迫文本处于大写或小写状态；

- ecUpperCase

TDBEdit 组件文本为大写状态。

- MaxLength

确定组件包含文本的最大长度。

- ReadOnly

确定 TDBEdit 组件中的文本是否可以编辑。

- OnChange

当编辑 TDBEdit 组件中的文本时触发该事件。

- OnKeyPress

当 TDBEdit 组件获得焦点时，按任意键触发该事件。

使用 TDBEdit 组件显示和编辑数据，需要通过数据源访问数据集数据，由于该组件只能显示一个字段的数据，在设置完数据源属性后，还需设置其所要显示的字段。

操作步骤如下。

（1）创建一个窗体，在窗体中放置 TDBEdit、TDataSource、TADOTable、TDBGrid 等组件，如图 7.2 所示。



图 7.2　供应商信息管理

（2）选中 TDBEdit 组件，在对象观察器中设置其 DataSource、DataField 属性，如图 7.3 所示。

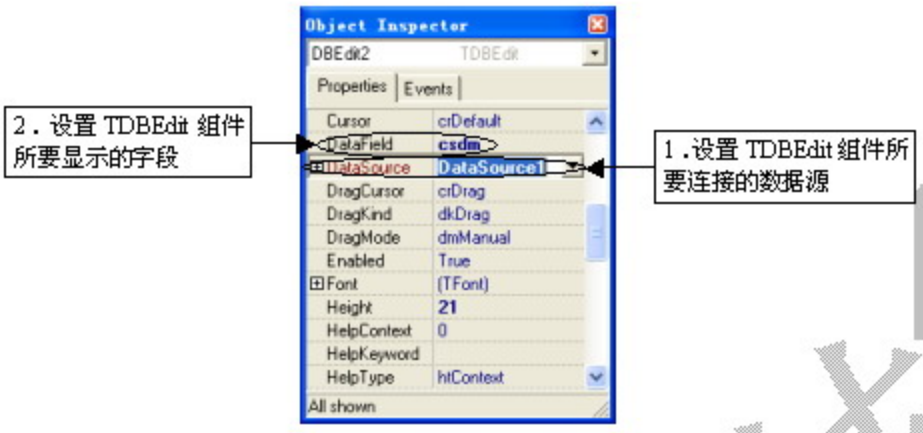


图 7.3　利用 TDBEdit 组件显示数据

注意：如果 TDBEdit 控件的 DataField 属性关联一个整型或浮点型的字段，则该编辑框中只允许能够转换为整型或浮点型数据的字符被键入。

7.3　TDBNavigator 组件介绍

TDBNavigator 组件提供了一组按钮，用来浏览、添加、修改、删除数据集。该组件如图 7.4 所示。

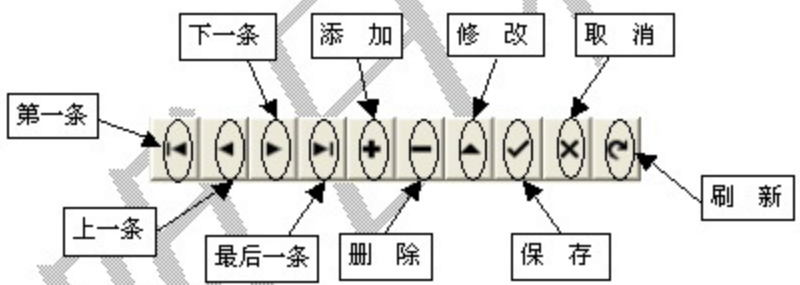


图 7.4　TDBNavigator 组件

TDBNavigator 组件主要属性如下。

- DataSource

该属性确定 TDBNavigator 组件所要连接的数据源。

- ConfirmDelete

该属性确定在删除操作时是否出现提示对话框。

- VisibleButtons

该属性用来设置 TDBNavigator 组件各按钮是否显示。

- ShowHint

该属性确定当鼠标在按钮上移动时，是否显示提示信息。

- Hints

当 ShowHint 属性为“True”时，Hints 属性中的字符串依次对应 TDBNavigator 组件按钮中的提示信息。

使用 TDBNavigator 组件，用户不需要任何 SQL 语句便可以实现数据的浏览、添加、修改、删除等操作。在使用该组件前，需要设置数据源属性，使其连接到指定的数据源上。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBEdit、TDataSource、TADOTable、TDBNavigator 等组件，如图 7.5 所示。

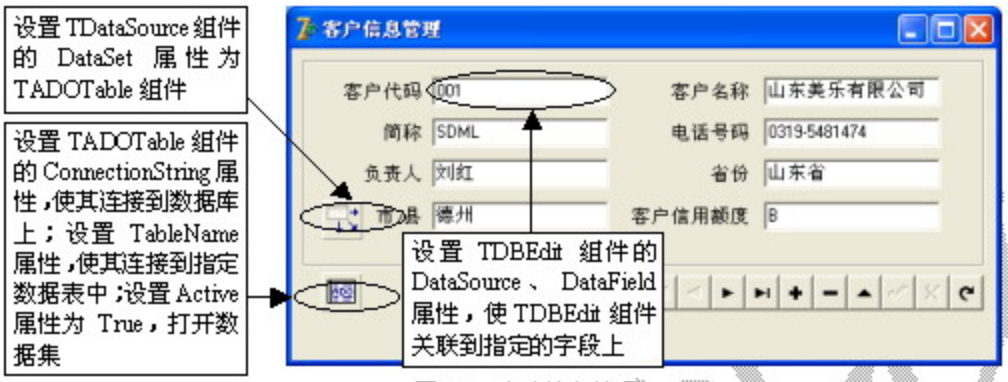


图 7.5　客户信息管理

（2）设置 TDBNavigator 组件的 DataSource 属性，使其指向 TDataSource 组件。运行程序，当用户单击“浏览”按钮时，TDBEdit 组件中会显示相应的记录数据。

> 注意：如果数据集当前指针处于首行，“第一条”、“上一条”按钮不可用；如果数据集当前指针处于尾行，“下一条”、“最后一条”按钮不可用。“保存”、“取消”按钮只有在添加或修改数据时才可用，而“修改”按钮在数据修改时不可用，当用户单击“保存”或“取消”按钮后才可用。

7.4　TDBImage 组件介绍

TDBImage 组件用来显示或提供修改数据集的大型二进制字段数据，使用该组件可以显示数据集对象中的图像。其主要属性、方法如下。

- DataSource

该属性确定 TDBImage 组件所要连接的数据源。

- DataField

确定 TDBImage 组件所要连接的字段，在设置该属性前，需要先设置 DataSource 属性。

- Center

该属性确定图像是否居中显示，如果为“False”，图像位于 TDBImage 组件的左上角。如果 Stretch 属性为“True”，该属性无效。

- Stretch

该属性为“True”，图像以 TDBImage 组件的尺寸和形状显示数据。

- Picture

该属性可以取得图形字段数据。

- CopyToClipBoard

将 Picture 属性指定的图像复制到剪贴板中。

- CutToClipBoard

删除 TDBImage 组件中的图像，将该图像复制到剪贴板中，调用该方法后，与 TDBImage 组件关联的字段值被设为空。

- LoadPicture

该方法将图像数据调入到 TDBImage 组件中。

- PasteFromClipBoard

将剪贴板中的图像数据赋给字段作为字段值。

TDBImage 组件主要用于显示数据库中存储的图像数据，在使用该组件前，需要设置数据源属性和图像字段属性。由于数据库中存储的图像尺寸可能大于 TDBImage 组件，致使 TDBImage 组件只能显示部分图像，为了将图像全部显示在 TDBImage 组件中，需要设置 TDBImage 组件的“Stretch”属性为“True”。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBEdit、TDataSource、TADOTable、TDBNavigator、TDBImage 等组件，如图 7.6 所示。

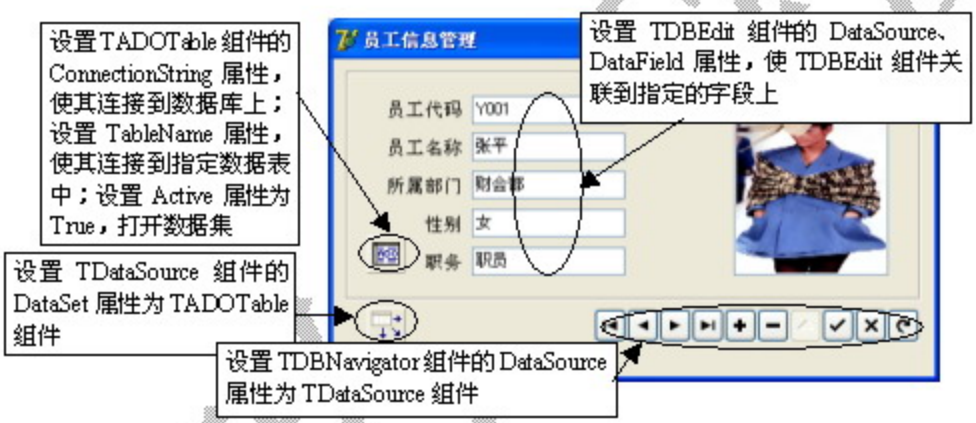


图 7.6　员工信息管理

（2）设置 TDBImage 组件属性，具体操作如图 7.7 所示。

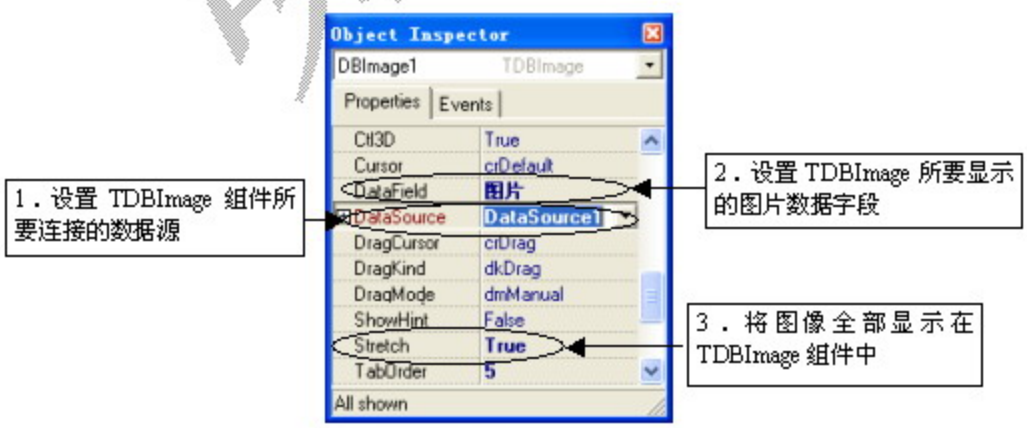


图 7.7　TDBImage 属性设置

7.5 TDBListBox 组件介绍

TDBListBox 组件用来显示或提供列表让用户通过列表选择输入数据。该组件与TListBox组件相似，只是提供了连接数据集的能力。其主要属性、方法如下。

- DataSource

该属性确定 TDBListBox 组件所要连接的数据源。

- DataField

确定 TDBListBox 组件所要连接的字段，在设置该属性前，需要先设置 DataSource 属性。

- Items

该属性列出了当前 TDBListBox 组件中的字符串，通过调用该属性的相关方法可以实现对 TDBListBox 组件中字符串的添加、插入、删除等操作。

- ItemHeight

该属性确定 TDBListBox 组件各字符串选项的高度。

- BorderStyle

确定列表框组件是否有边框。

- IntegralHeight

该属性确定是否允许列表框中的项目部分被显示。该属性为“True”时，不允许列表框中的项目被部分显示，“IntegralHeight”属性为“False”时，列表框的高度不受限制，但是列表框中最后的项目可能不会被全部显示。当“Style”属性为“lbOwnerDrawVariable”时，设置“IntegralHeight”属性没有任何影响。

- ItemIndex

该属性是指选项在列表中的位置值。在列表中，第一项的位置值为 0，第二项为 1，依次类推。

TDBListBox 组件提供了列表供用户选择输入数据，当数据集移动时，与 TDBListBox 组件相关联的字段若存在于列表中，则列表中第一个符合的项目会被选中，处于高亮显示。当数据集允许修改时，改变列表中的选项会更新数据集中相关字段值。在使用 TDBListBox 组件前，需要设置“DataSource”属性与“DataField”属性。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBEdit、TDataSource、TADOTable、TDBNavigator、TDBListBox 等组件，具体操作如图 7.8 所示。

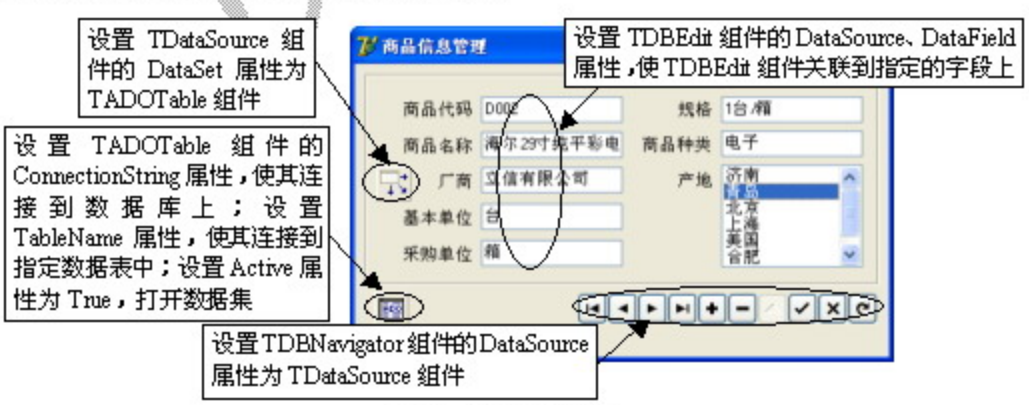


图 7.8 商品信息管理

（2）设置 TDBListBox 组件的 DataSource、DataField、Items 属性，具体操作如图 7.9 所示。

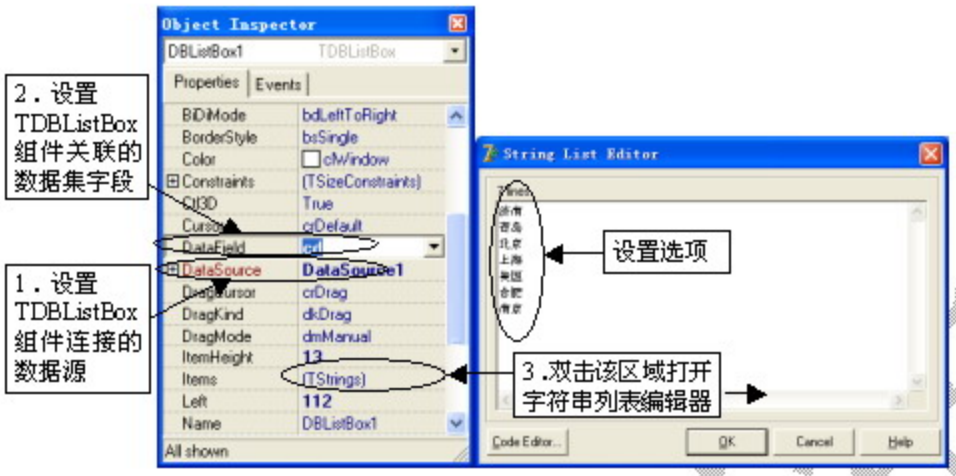


图 7.9　设置 TDBListBox 属性

在设置完 TDBListBox 组件的相关属性后，当移动数据集记录时，如果 TDBListBox 列表中存在其关联字段的数据时，则该选项会被高亮显示。

7.6　TDBComboBox 组件介绍

TDBComboBox 组件用来显示或提供下拉式菜单，让用户可以通过下拉菜单选择数据，特点是占用的屏幕空间少。TDBComboBox 组件的主要属性如下。

- DataSource

该属性确定 TDBComboBox 组件所要连接的数据源。

- DataField

确定 TDBComboBox 组件所要连接的字段，在设置该属性前，需要先设置 DataSource 属性。

- AutoComplete

该属性确定在编辑框中输入一个字符是否将与该字符匹配的选项显示出来。

- AutoDropDown

该属性确定组合框获得焦点时，用户按键，下拉列表是否自动显示。

- DropDownCount

该属性确定下拉列表中一次可显示的字符串个数。如果列表中的选项比该属性值大，下拉列表会出现滚动条，如果列表中的项目比该属性值小，下拉列表会收缩来容纳选项。

- ItemIndex

该属性指明当前项在组合框中的位置。如果组合框中没有选项，该值为-1，否则列表框中第一项的“ItemIndex”值为 0，第二项为 1，依次类推。

使用 TDBComboBox 组件，用户可以通过下拉式菜单选择输入数据，当数据集记录移动时，在组合框中会显示 TDBComboBox 组件关联的字段值，当记录集允许修改时，改变组合框中的当前选项，会直接更新数据集的字段值。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBEdit、TDBGrid、TDataSource、TADOTable、TDBComboBox 等组件，具体操作如图 7.10 所示。

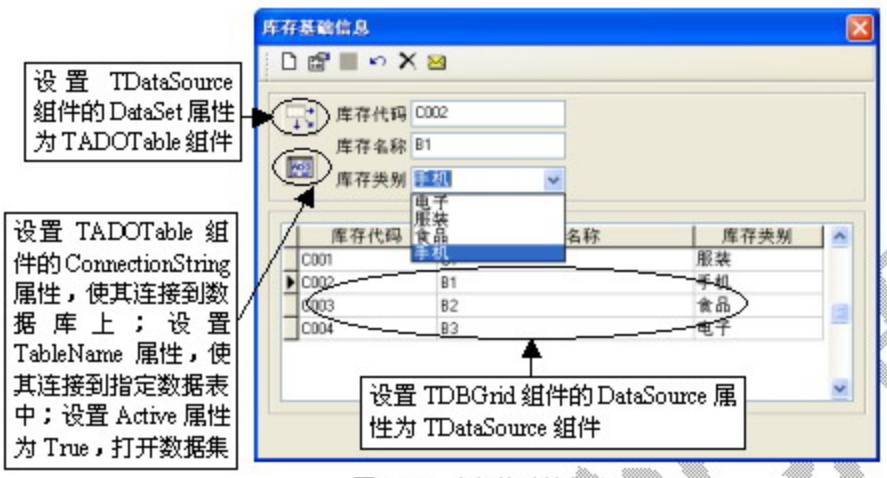


图 7.10　库存基础信息

（2）设置 TDBComboBox 组件的 DataSource、DataField、Items 属性。在组合框中改变当前选项，当移动记录或提交数据时，程序会自动保存所做的修改。

7.7　TDBLookupListBox 组件介绍

TDBLookupListBox 组件用来显示另一个数据集相对应的字段值，其主要属性、方法如下。

- DataSource

该属性指定 TDBLookupListBox 组件与哪个数据源变动。

- DataField

该属性指定 TDBLookupListBox 组件关联到指定数据源的哪个字段。

- ListSource

该属性提供用来显示字段数据的数据源。

- ListField

该属性指定 ListSource 显示的字段名称，通过该属性可以设置多个字段，字段之间用分号分隔。

- RowCount

设置查找列表框中显示的行数。

- SelectedItem

确定用户使用查找列表框时选择的值。如果没有项目被选中，该属性返回一个空字符串。

- KeyField

将 TDBLookupListBox 组件的 ListSource 属性设置的数据源连接到 DataSource 属性指定的数据源。

- KeyValue

设置该属性，在 ListSource 指定的数据源中查找与其匹配的一个记录。

TDBLookupListBox 组件为填充字段提供了一个查找项列表，列表中的数据来自于另一个数据集中某个字段值。使用该组件可以利用另一个数据集中字段值设置当前字段值。在使用 TDBLookupListBox 组件前，需要对其属性进行设置。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBGrid、TDataSource、TADOTable、TDBLookupListBox 等组件，具体操作如图 7.11 所示。

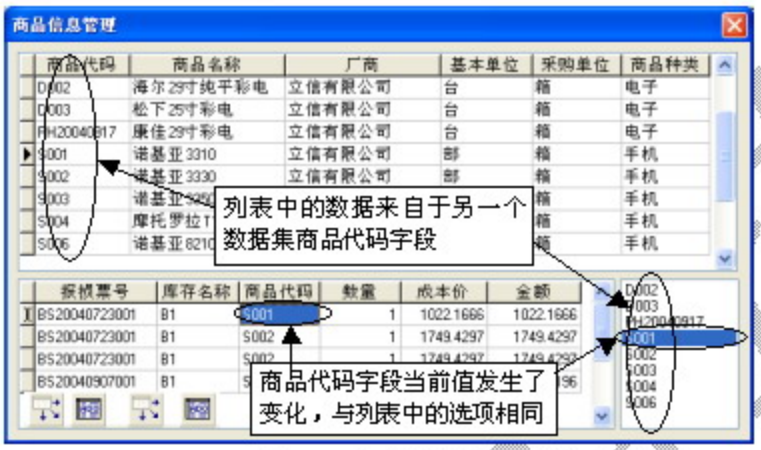


图 7.11　商品信息管理

（2）分别设置两个 TADOTable 组件的 ConnectionString、TableName 属性，使其连接到数据表中，将 Active 属性设置为 True，打开数据集。

（3）设置两个 TDataSource 组件的 DataSet 属性，分别指向 TADOTable 组件。

（4）设置 TDBLookupListBox 组件属性，具体操作如图 7.12 所示。

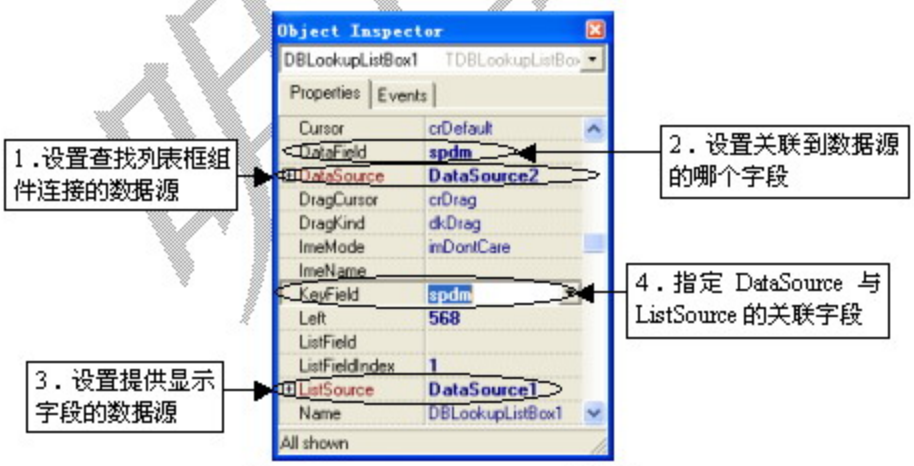


图 7.12　TDBLookupListBox 属性设置

在设置完 TDBLookupListBox 组件的属性后，当“ListSource”所访问的数据集处于激活状态时，“KeyField”属性指定的字段将显示在列表框中；当“DataSource”所访问的数据集处于激活状态，并且设置了“DataField”属性时，用户在列表中选择一个选项，该选项数据将出现在“DataField”属性指定的字段中（当前记录）。

7.8 TDBRadioGroup 组件介绍

TDBRadioGroup 组件是连接到数据库的一组单选按钮，当该组件上的单选按钮被选中时，被选中按钮的值会成为数据集内当前行字段的内容，如果单选按钮没有值，会将单选按钮的标题作为它的值。TDBRadioGroup 组件的主要属性、方法如下。

- DataSource

该属性确定 TDBRadioGroup 组件所要连接的数据源。

- DataField

确定 TDBRadioGroup 组件所要连接的字段，在设置该属性前，需要先设置 DataSource 属性。

- Items

通过该属性可以设置单选按钮的标题属性。

- Values

该属性确定单选按钮的值。

- Columns

该属性确定单选按钮分组框中的列数。

TDBRadioGroup 组件能够将一批具有数据感知功能的单选按钮组合成一组，当该组件关联到数据集的某个字段时，改变数据集当前记录，如果字段值在 TDBRadioGroup 组件的单选按钮中有对应的值，相应的单选按钮将处于选中状态；如果更改当前选中的单选按钮，数据集中关联的字段会发生相应的变化，此时移动数据集记录或提交数据会保存所做的修改。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBGrid、TDataSource、TADOTable、TDBEdit、TDBRadioGroup 等组件，如图 7.13 所示。



图 7.13 付款代码

（2）分别设置两个 TADOTable 组件的 ConnectionString、TableName 属性，使其连接到数据表中，将 Active 属性设为 True，打开数据集。

（3）设置 TDataSource 组件的 DataSet 属性，使其指向 TADOTable 组件。

（4）设置 TDBRadioGroup 组件属性，具体操作如图 7.14 所示。

注意：如果 TDBRadioGroup 组件中的选项过多，可以设置 Columns 属性，使其可以多列显示。

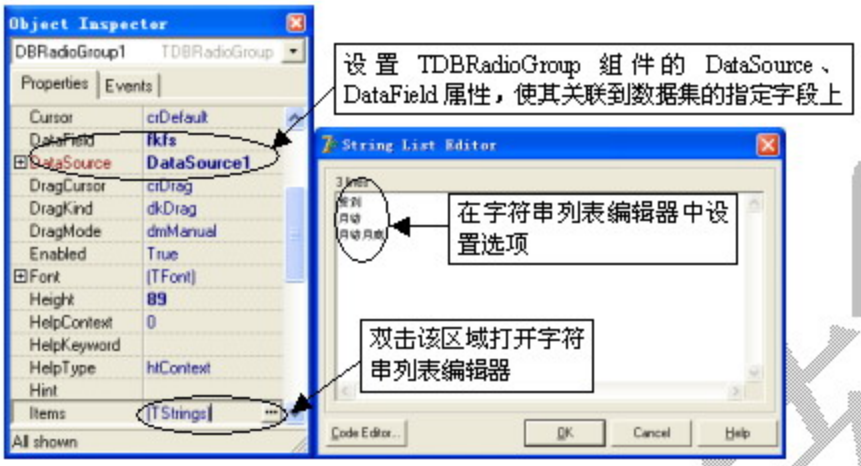


图 7.14　TDBRadioGroup 属性设置

7.9　TDBGrid 组件介绍

TDBGrid 组件是数据控制组件中应用最多的组件，该组件以表格形式显示数据表中的记录。它可以与 TDBNavigator 组件结合使用，在程序运行时，用户可以通过 TDBNavigator 组件来移动、添加、删除表格中的数据。TDBGrid 组件的主要属性、事件如下。

- Align

设置组件在父类中的对齐方式。

- DataSource

设置该组件连接的数据源。

- Columns

确定该组件所显示的数据列。

- Options

该属性是一个集合类型，用来设置 TDBGrid 组件的行为和外观，共有 13 个元素，各意义如下。

- dgEditing

用户可以编辑表格，如果 Options 属性包含 dgRowSelect，dgEditing 将被忽略。

- dgAlwaysShowEditor

表格总处于编辑状态，只有在 Options 属性包含 dgEditing 时，dgAlwaysShowEditor 才有意义，如果 Options 属性包含 dgRowSelect，dgAlwaysShowEditor 将被忽略。

- dgTitles

显示字段标题。

- dgIndicator

在当前行的首列显示一个小指针。

- dgColumnResize

字段能够调整大小并且能够移动。

- dgColLines

网格显示垂直线。

- dgRowLines

网格显示水平线。

- dgTabs

用户可以按 Tab 键或 Shift+Tab 键浏览数据。

- dgRowSelect

用户能够选中整行数据，如果 Options 中包含 dgRowSelect，dgEditing 与 dgAlwaysShowEditor 将被忽略。

- dgAlwaysShowSelection

当网格没有获得焦点时，被选中的单元格也会显示焦点区域。

- dgConfirmDelete

当用户在网格中删除一行时，会出现提示消息框。

- dgCancelOnExit

在添加记录时，当用户离开网格且记录未改变时，会放弃添加操作，该选项防止不小心向数据表中插入空记录。

- dgMultiSelect

一次可以选中多行记录。

- OnCellClick

当用户单击单元格时触发该事件。

- OnColEnter

当焦点移到一个新的单元格时触发该事件。

- OnColExit

当单元格失去焦点时触发该事件。

- OnColumnMoved

当网格列被移动时触发该事件。

TDBGrid 组件以表格形式显示和处理数据集记录，用户可以在表格中插入、编辑、删除数据。在程序运行时，可以通过 TDBNavigator 组件在网格中移动数据，用户在表格中编辑数据，只有在更改当前记录或关闭应用程序时才会保存所做的修改。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBGrid、TDataSource、TADOTable、TDBNavigator 等组件，如图 7.15 所示。



图 7.15　入库信息

（2）分别设置两个 TADOTable 组件的 ConnectionString、TableName 属性，使其连接到

数据表中，将 Active 属性设为 True，打开数据集。

（3）设置 TDataSource 组件的 DataSet 属性，使其指向 TADOTable 组件；设置 TDBNavigator 组件的 DataSource 属性为 TDataSource 组件。

（4）设置 TDBGrid 组件属性，具体操作如图 7.16 所示。

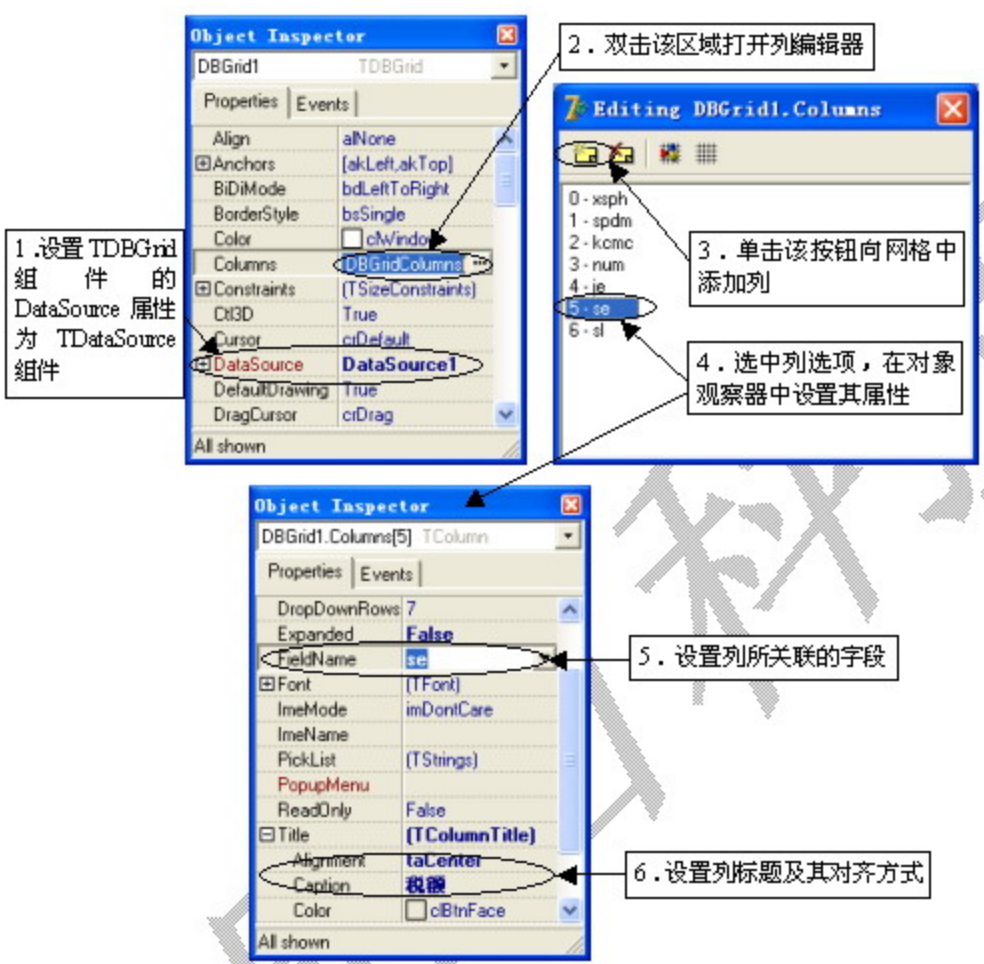


图 7.16　TDBGrid 属性设置

注意：在使用 TDBGrid 组件显示数据时，如果只是为了显示数据，而不想修改数据，建议将该组件的 ReadOnly 属性设为 True，否则用户按 Ctrl+Delete 组合键将删除当前行数据。

7.10　TDBCtrlGrid 组件介绍

TDBCtrlGrid 组件用于在自由版式设计的网格中显示数据源对象中的记录，它可以以一种比标准数据库网格（TDBGrid）更为美观的形式查看数据库中的字段，其主要属性、方法如下。

- AllowDelete

确定用户是否可以删除记录。

- AllowInsert

确定用户是否可以向数据库控制网格中插入或追加新记录。

- ColCount

指定表格中显示的面板列。

- PanelCount

指定面板数。

- PanelWidth

指定面板的像素宽度。

- RowCount

指定面板的行数。

- SelectedColor

指定数据集当前记录面板的背景色。

- DataSource

指定该组件所要连接的数据源。

TDBCtrlGrid 组件可以显示和编辑数据表或查询对象中的记录，它与 TDBGrid 组件不同，TDBGrid 组件将每条记录显示在单独的一行中，而 TDBCtrlGrid 组件将每条记录显示在自己的面板上，在程序设计期设计一个面板，TDBCtrlGrid 组件会将每条显示的记录复制到这个面板中。

操作步骤如下。

（1）新建一个窗体，在窗体中放置 TDBCtrlGrid、TDataSource、TADOTable、TDBNavigator、TDBEdit 等组件，如图 7.17 所示。



图 7.17　商品销售明细

（2）分别设置两个 TADOTable 组件的 ConnectionString、TableName 属性，使其连接到数据表中，将 Active 属性设为 True，打开数据集。

（3）设置 TDataSource 组件的 DataSet 属性，使其指向 TADOTable 组件；设置 TDBNavigator 组件的 DataSource 属性为 TDataSource 组件；设置 TDBCtrlGrid 组件的 DataSource 属性为 TDataSource 组件；设置 TDBEdit 组件的 DataSource 属性为 TDataSource 组件，DataField 属性为相关数据集字段。

（4）运行程序，结果如图 7.18 所示。



图 7.18　商品销售明细运行结果

明日科技

报表与打印程序设计

第 8 章

8.1 利用 Rave 组件设计报表

8.1.1 常用 Rave 组件介绍

1. TRvProject 组件

- Active

该属性确定 TRvProject 组件是否处于活动状态。

- DLLFile

指定“LoadDesigner”属性为“True”时使用的 DLL 文件。

- Engine

当通过 TRvProject 组件打印报表时，该属性允许用户定义一个报表引擎。

- LoadDesigner

该属性为“True”，会加载“DLLFile”属性指定的终端用户文件。

- ProjectFile

该属性指定一个报表项目文件，当 TRvProject 组件打开时会加载该文件。

- RaveBlobDateTime

返回报表项目最后一次被应用程序窗体加载的日期时间值。

- ReportDesc

返回当前被选中报表的描述信息。

- ReportFullName

返回当前被选中报表的 FullName 信息。

- ReportName

返回当前被选中报表的 Name 信息。

❑ StoreRAV

该属性返回被存储的报表项目是否可执行。

❑ ClearRaveBlob

该方法清除当前应用程序窗体中加载的报表项目。

❑ Close

该方法关闭报表项目文件，同时释放内存，如果用户在应用程序中调用了 RvProject 组件的“Open”方法，则应在程序结束前调用“Close”方法。

❑ Execute

该方法开始打印当前被选中的报表项目。

2. TRvSystem 组件

❑ BaseReport

通过该属性可以访问 TRvSystem 组件创建的 TBaseReport（所有输出组件的基类）对象。在“Execute”方法被调用前该属性值为空。在通常情况下没有必要访问该属性，因为对于所有的打印事件，TBaseReport 对象被作为“Sender”参数传递。

❑ DefaultDest

确定系统默认的输出设备，其属性值可以为“rdFile”、“rdPreview”或“rdPrinter”。

❑ GridHoriz

确定屏幕预览时网格水平方向的大小。

❑ GridPen

指定通过“GridVert”和“GridHoriz”属性绘制的网格画笔。

❑ OutputFileName

标识报表被输出的文件名。

3. TRvDataSetConnection 组件

❑ DataSet

标识 TRvDataSetConnection 组件使用的数据集组件。

❑ FieldAliasList

该属性可以为指定的字段设置一个别名。

❑ LocalFilter

该属性确定是依靠数据连接组件过滤还是依靠数据库进行过滤。

❑ RuntimeVisibility

该属性确定数据连接对于终端用户的可见度，共有 3 个属性值，意义如下。

- rtNone

运行时对于外部的应用程序不可见。

- rtDeveloper

运行时只对开发者可见。

- rtEndUser

对于任何终端用户都可见。

8.1.2 使用 Rave 组件设计报表

（1）新建一个应用程序，在窗体中放置 TTable 组件、TRvDataSetConnection 组件、

TRvProject 组件、TButtton 组件。

（2）设置 TTable 组件的“DatabaseName”属性、“TableName”属性，使其连接到数据表中，设置“Active”属性为“True”，打开与数据表的连接。

（3）设置 TRvDataSetConnection 组件的“DataSet”属性为“Table1”（TTable 组件）。

（4）双击 TRvDataSetConnection 组件的“FieldAliasList”属性，打开字符串列表编辑器，为数据表字段设置别名，如图 8.1 所示。

（5）双击 TRvProject 组件，打开 Rave 报表工具，如图 8.2 所示。

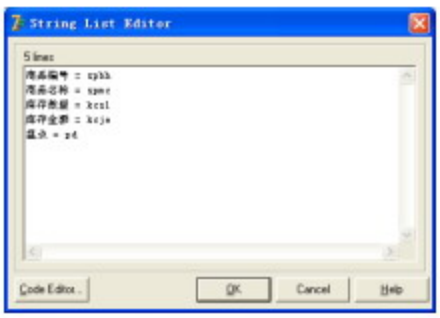


图 8.1　字符串列表编辑器

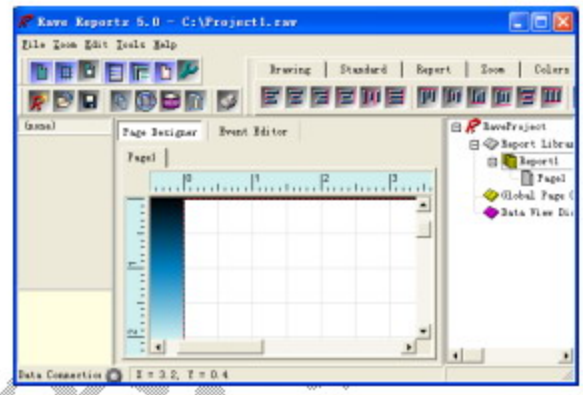


图 8.2　Rave 报表工具

（6）单击“File”/“New Data Object”菜单项，打开数据连接对话框，如图 8.3 所示。

（7）选中“Direct Data View”选项，单击 Next 按钮切换至激活数据连接页面，如图 8.4 所示。



图 8.3　数据连接对话框

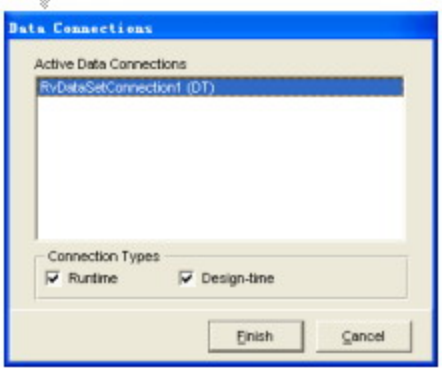


图 8.4　激活数据连接

（8）单击 Finish 按钮完成数据视图的创建。

（9）单击“Tools”/“Report Wizards”/“Simple Table”菜单项，打开“Simple Table”对话框，如图 8.5 所示。

（10）单击 Next 按钮切换至选择字段页面，如图 8.6 所示。

（11）单击 All 按钮选中所有字段，单击 Next 按钮切换至调整字段顺序页面，如图 8.7 所示。

（12）单击 Next 按钮切换至设计报表版面页面，如图 8.8 所示。

（13）在“Report Title”编辑框中输入“我的第一个报表”，单击 Next 按钮切换至改变字体页面，如图 8.9 所示。

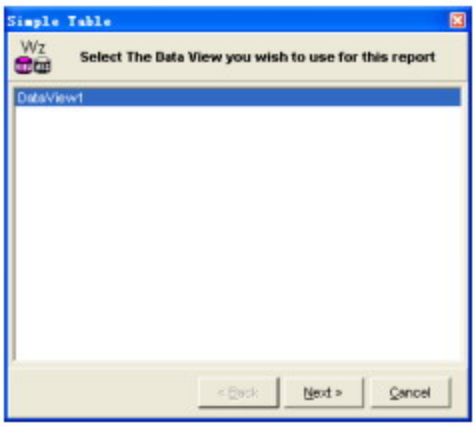


图 8.5　Simple Table 对话框

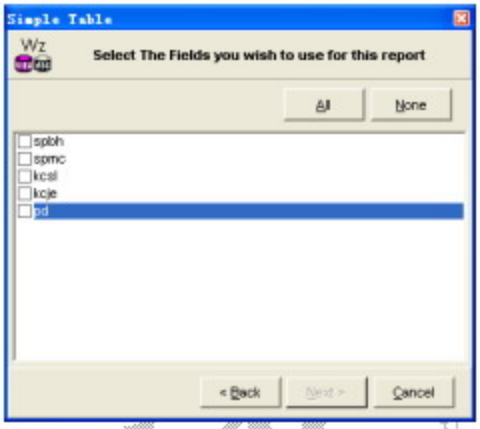


图 8.6　选择字段

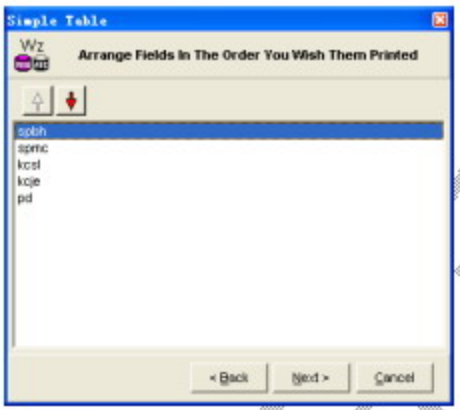


图 8.7　调整字段顺序

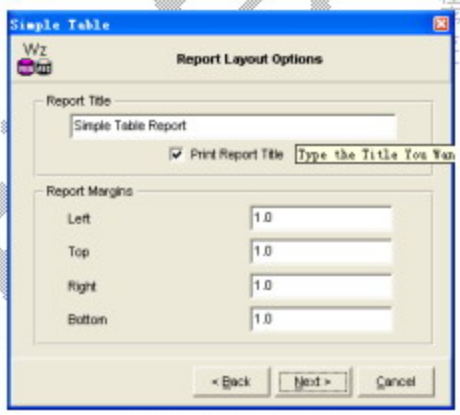


图 8.8　设计报表页面



图 8.9　改变字体

（14）单击 Generate 按钮生成报表，如图 8.10 所示。

（15）在报表页面中更改字段标题，选中 TText 组件，在属性列表中修改其“Text”属性，如图 8.11 所示。

（16）按 F9 键查看预览结果，如图 8.12 所示。

（17）单击“File”/“Save”菜单项，保存报表为“Project1.rav”，存放在 C 盘根目录下。

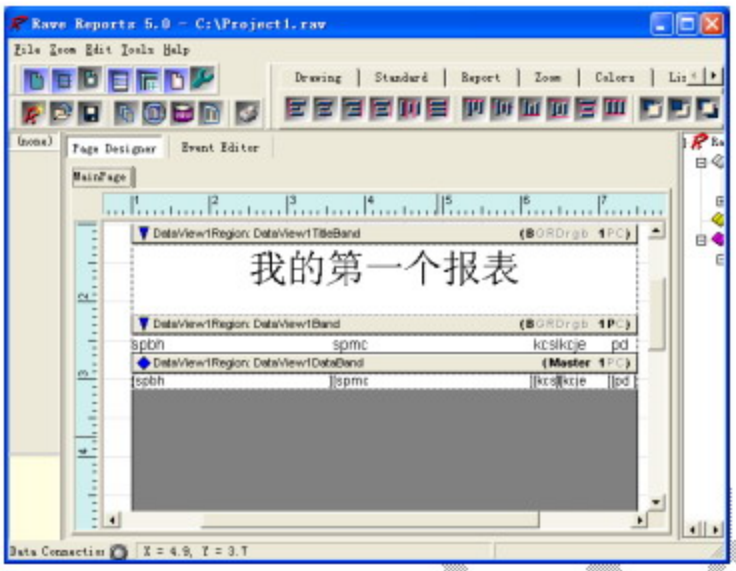


图 8.10　报表页面

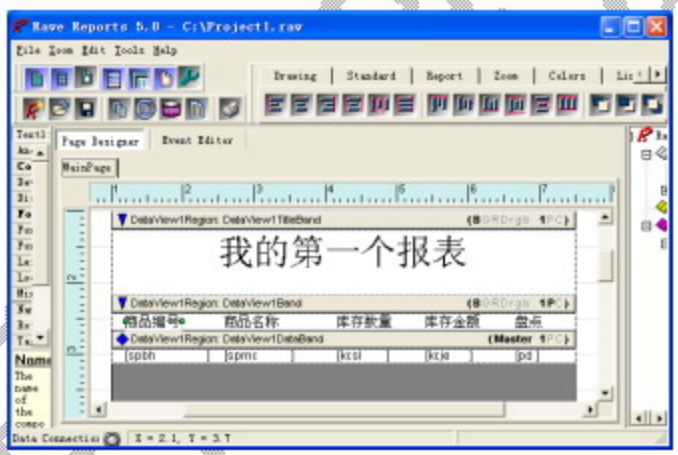


图 8.11　报表页面效果图

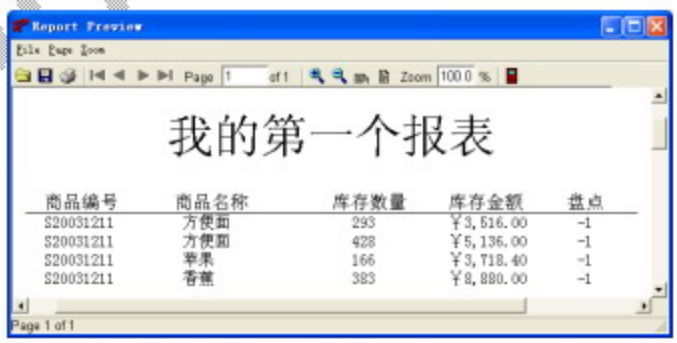


图 8.12　报表预览结果

（18）关闭 Rave 报表工具，在对象观察器中设置 TRvProject 组件的“Project File”属性为“C:\Project1.rav”。

（19）处理 TButton 组件的单击事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  RvProject1.Execute;
end;
```

（20）运行程序，单击按钮会打开报表，如图 8.13 所示。

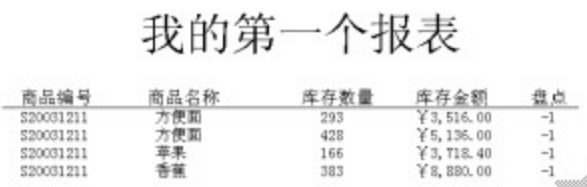


图 8.13　库存报表

8.1.3　使用代码创建报表

（1）新建一个应用程序，在窗体中放置 TButton 组件、TRvSystem 组件。

（2）处理 TRvSystem 组件的 OnPrint 事件，代码如下：

```
procedure TForm1.RvSystem1Print(Sender: TObject);
var
  Row: Integer;
begin
  with Sender as TBaseRePort do
  begin
    YPos := 0.5; //设置数据在页面中打印的纵坐标
    SetFont('宋体',18); //设置字体
    Underline := True; //添加下划线
    FontColor := clRed; //设置字体颜色
    PrintCenter('我的第一个报表',PageWidth/2); //打印报表标题
    SetFont('宋体',10);
    FontColor := clBlue;
    YPos := YPos + 0.5;
    SetTab(2.2,pjCenter,1,0,0,0);
    SetTab(na,pjCenter,1,0,0,0);
    SetTab(na,pjCenter,1,0,0,0);
    SetTab(na,pjCenter,1,0,0,0);
    Tab(-2,Na,-2,Na,10);
    Print('语文');
    Tab(-2,Na,-2,Na,10);
    Print('数学');
    Tab(-2,Na,-2,Na,10);
    Print('英语');
    Tab(-2,-2,-2,Na,10);
    Print('政治');
    SetFont('宋体',8);
    FontColor := clBlue;
    YPos := YPos +0.15;
    SetTab(2.2,pjCenter,1,0,0,0);
```

```
        SetTab(na,pjCenter,1,0,0,0);
        SetTab(na,pjCenter,1,0,0,0);
        SetTab(na,pjCenter,1,0,0,0);
        For row := 1 to 4 do
        begin
          Tab(-2,Na,-2,-2,10);
          Print('90');
          Tab(-2,Na,-2,-2,10);
          Print('85');
          Tab(-2,Na,-2,-2,10);
          Print('62');
          Tab(-2,-2,-2,-2,10);
          Print('77');
          Println('');
        end;
      end;
    end;
```

（3）处理按钮的 OnClick 事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  RvSystem1.Execute;
end;
```

（4）运行程序，单击按钮，报表预览结果如图 8.14 所示。



图 8.14　报表预览结果

8.2　特色报表设计

8.2.1　设计页眉和页脚

（1）新建一个应用程序，在窗体中放置 TTable 组件，设置该组件的“DatabaseName”、“TableName”属性，设置“Active”属性为“True”。

（2）在窗体上放置一个 TRvDataSetConnection 组件，设置“DataSet”属性为“Table1”。

（3）单击“Tools”/“Rave Designer”菜单项打开 Rave 报表工具，按照 8.1.2 节步骤

（5）～步骤（8）的介绍，创建一个数据视图对象 DataView1。

（4）在报表区域中放置一个 TRegion 组件，在 TRegion 组件中放置 TBand 组件与 TDataBand 组件各一个。在属性编辑器中设置 TDataBand 组件的属性为“DataView1”。

（5）在 TBand 组件中放置 TLine 组件与 TText 组件，在 TDataBand 组件中放置 TDataText 组件，设置 TDataText 组件的“DataView”属性为“DataView1”，在“DataField”属性的下拉列表框中选择数据字段。

（6）在 TRegion 组件的右下角放置 TText 组件与 TLine 组件，用来设置报表页脚。报表界面如图 8.15 所示。



图 8.15　报表界面

（7）按 F9 键预览报表，结果如图 8.16 所示。



图 8.16　报表预览结果

（8）单击“File”/“Save”菜单项将报表保存为“Project1.rav”。

8.2.2　设计带表格的报表

在 Rave 报表工具的 Drawing 选项卡中提供了各种绘制表格的组件，使用这些组件可以在报表页面设计各种表格，下面以向 8.2.1 节中的报表添加表格为例介绍如何设计表格。

（1）打开 8.2.1 节设计的报表“Project1.rav”，在 TBand 组件中放置 THLine 组件与 TVLine 组件。

（2）调整 THLine 组件与 TVLine 组件的宽度和高度，报表版面如图 8.17 所示。



图 8.17　设计表格

（3）按 F9 键预览报表，如果表格出现断续或一些行没有表格线，请调整 TBand 组件与 TDataBand 组件的高度，效果如图 8.18 所示。

长春明日科技有限公司

员工信息报表

员工编号	员工姓名	性别	年龄	职务	工资
Y001	张震	男	25	部门经理	￥2,300.00
Y002	李平	女	20	职员	￥800.00
Y003	周亮	男	23	开发人员	￥1,500.00
Y004	赵丽	女	22	开发人员	￥1,000.00
Y005	孙军	男	24	销售经理	￥2,000.00

图 8.18　表格报表

8.2.3　打印分栏文本

（1）新建一个应用程序，在窗体中放置 TTable 组件，设置该组件的“DatabaseName”、“TableName”属性，设置“Active”属性为“True”。

（2）在窗体上放置一个 TRvDataSetConnection 组件，设置“DataSet”属性为“Table1”。

（3）单击“Tools”/“Rave Designer”菜单项打开 Rave 报表工具，创建一个数据视图对象 DataView1。

（4）在报表区域中放置一个 TRegion 组件，在 TRegion 组件中放置 TBand 组件与 TDataBand 组件各一个。在属性编辑器中设置 TDataBand 组件的属性为“DataView1”，设置“Columns”属性为“2”。

（5）在 TBand 组件中放置一个 TText 组件，用来设置报表标题。

（6）在 TDataBand 组件中放置 TText 组件、TDataText 组件、TLine 组件、TBitmap 组件，设置 TDataText 组件与 TBitmap 组件的“DataView”属性为“DataView1”，在“DataField”属性下拉列表框中选择相应的字段。

（7）报表界面如图 8.19 所示。



图 8.19　分栏报表界面

（8）按 F9 键预览报表，效果如图 8.20 所示。



图 8.20　分栏报表

8.3 使用 Rave 组件设计报表实例应用

8.3.1 设计主从式报表

主从式报表是在一个报表中同时显示主表与从表中的数据，在设计主从式报表时，Dephi 应用程序中需要有两个数据集，如果使用 TRave 组件设计主从式报表，不需要在程序中设定数据集的主从关系，下面结合具体实例介绍如何设计主从式报表。

（1）新建一个应用程序，在窗体中放置两个 TTable 组件，分别设置 TTable 组件的“DatabaseName”属性、“TableName”属性，使其连接到相应的数据表中，将“Active”属性设为“True”，打开与数据表的连接。

（2）在窗体中放置两个 TRvDataSetConnection 组件，分别设置“DataSet”属性为相应的 TTable 组件。

（3）单击“Tools”/“Rave Designer”菜单项打开 Rave 报表工具，分别建立两个数据视图对象 DataView1 与 DataView2，其数据连接对象分别为相应的 TRvDataSetConnection 组件。

（4）在报表区域中放置一个 TRegion 组件，在 TRegion 组件中放置两个 TBand 组件与两个 TDataBand 组件，其中 TBand 组件用来设置报表的标题与摘要，TDataBand 组件用来显示主表与从表中的数据。

（5）在 Band1 组件中放置 3 个 TText 组件，设置 TText 组件的“Text”属性以显示报表标题与操作员信息。

（6）在 Band2 组件中放置两个 TText 组件，设置其中一个 TText 组件的“Text”属性为“时间”，选中另一个 TText 组件，在报表中打开“Event Editor”选项卡，如图 8.21 所示。

（7）在“Available Events”下拉列表框中选择“OnBeforePrint”事件，在“Defined Events”编辑框中会显示用户当前选择的事件，如图 8.22 所示。

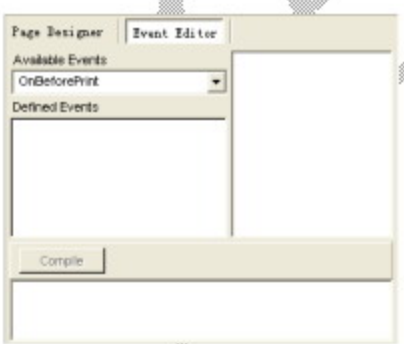


图 8.21 事件编辑器 1

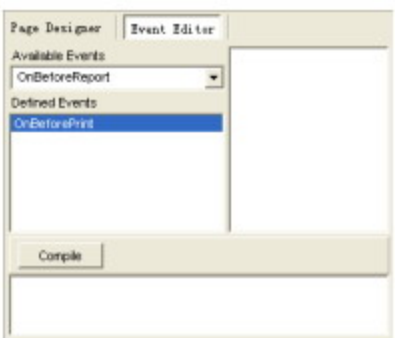


图 8.22 事件编辑器 2

（8）在 Compile 按钮下方的编辑框中输入程序代码：

```
Text13.Text := DateTimetoStr(Date);
```

（9）单击 Compile 按钮编译代码。

（10）设置 DataBand1 组件的“DataView”属性为“DataView1”，在属性列表中双击“BandStyle”属性，打开边条风格编辑器，如图 8.23 所示。

（11）在打开的边条编辑器中只选中“First”复选框，其他复选框处于非选中状态。

（12）在属性编辑器中设置 DataBand2 组件的“DataView”属性为“DataView2”，设

置“ControllerBand”属性为“DataBand1”，在“DetailKey”属性下拉列表中选择一个与主表关联的字段，在“MasterDataView”属性列表中选择“DataView1”，在“MasterKey”属性列表中选择主表的关联字段，通常情况下，“MasterKey”属性值与“DetailKey”属性值相同。

（13）在属性编辑器中双击 DataBand2 组件的“BandStyle”属性，打开边条风格编辑器，如图 8.24 所示。

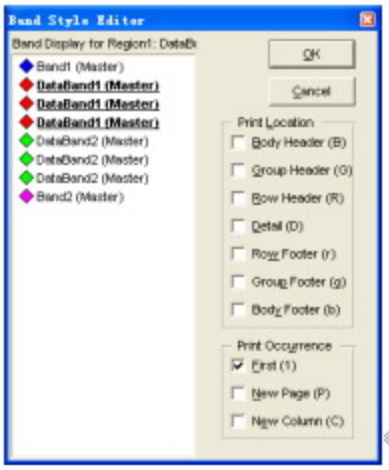


图 8.23　边条风格编辑器 1

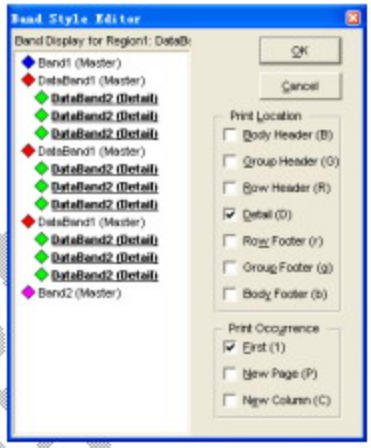


图 8.24　边条风格编辑器 2

（14）在边条风格编辑器中选中“Detail”复选框与“First”复选框，其他复选框处于非选中状态。

（15）在 TDataBand 组件中放置 TText 组件、TDataText 组件与 THLine 组件，设置 TDataText 组件的“DataView”属性与“DataField”属性，报表版面如图 8.25 所示。

（16）按 F9 键预览报表，如图 8.26 所示。



图 8.25　主从式报表界面

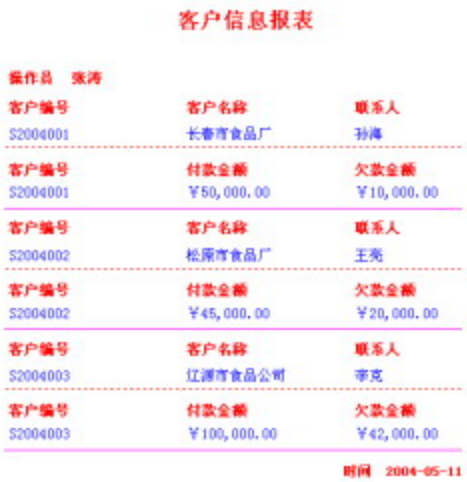


图 8.26　主从式报表

8.3.2　设计汇款单式报表

（1）新建一个应用程序，在窗体中放置 TTable、TRvDataSetConnection 组件，设置

TTable组件的“DatabaseName”、“TableName”属性使其连接到数据表中，将“Active”属性设为“True”，打开与数据表的连接。设置TRvDataSetConnection组件的“DataSet”属性为“Table1”。

（2）单击“Tools”/“Rave Designer”菜单项打开Rave报表工具，新建一个数据视图DataView1。在报表中放置一个TRegion组件，在TRegion组件中放置一个TDataBand组件，设置TDataBand组件的“DataView”属性为“DataView1”。

（3）在TDataBand组件中放置TText组件、TSquare组件、TVLine组件、TDataText组件、TMemo组件、THLine组件。设置TDataText组件的“DataView”、“DataField”属性，使其连接到指定的字段中。报表版面如图8.27所示。

（4）按F9键预览报表，如图8.28所示。



图8.27 汇款单报表界面



图8.28 汇款单式报表

8.3.3 设计分组式报表

（1）新建一个应用程序，在窗体上放置TTable、TRvDataSetConnection等组件，设置TTable组件的“DatabaseName”属性与“TableName”属性，使其连接到数据表上，将“Active”属性设置为“True”，打开与数据表的连接。

（2）单击“Tools”/“Rave Designer”菜单项打开Rave报表工具，新建一个数据视图DataView1。在报表中放置一个TRegion组件，在TRegion组件中放置两个TBand组件，显示报表标题与统计信息，放置一个TDataBand组件，设置TDataBand组件的“DataView”属性为“DataView1”以显示数据视图中的数据。

（3）在Band1组件中放置THLine组件、TText组件，选中一个TText组件，在报表中单击“Event Editor”选项卡切换至事件编辑器页面，处理TText组件的“OnBeforePrint”事件，具体步骤请参考8.3.1节步骤（6）～步骤（9），设置其他TText组件的“Text”属性以显示报表标题及其操作员、字段名称信息。

（4）在TDataBand组件中放置THLine组件、TDataText组件，设置TDataText组件的“DataView”属性与“DataField”属性，使其连接到指定的字段中。

（5）在Band2组件中放置TText组件、TCalcText组件，设置TCalcText组件的“DataView”属性为“DataView1”，设置“CalcType”属性为“ctAverage”，设置“Controller”属性为“DataBand1”，在“DataField”属性下拉列表中选择要统计的数据字段。报表界面如图8.29所示。

（6）按F9键预览报表，结果如图8.30所示。



图 8.29　分组报表界面

员工信息报表

操作员 王丽　　时间 2004-05-12

员工编号	员工姓名	性别	年龄	职务	工资
Y001	张震	男	25	部门经理	2300
Y002	李平	女	20	职员	800
Y003	周亮	男	23	开发人员	1500
Y004	赵丽	女	22	开发人员	1000
Y005	孙军	男	24	销售经理	2000

平均年龄 22.8　　平均工资 1520

图 8.30　分组式报表

8.3.4　设计随查询结果自动更新的报表

（1）新建一个应用程序，在窗体中放置 TPanel、TLabel、TSpeedButton、TBitBtn、TComboBox、TDBGrid、TDataSource、TQuery、TRvProject、TRvDataSetConnection 组件，界面如图 8.31 所示。

（2）设置 BitBtn1 的“Enabled”属性为“False”，设置 BitBtn2 的“Kind”属性为“bkClose”，设置 DBGrid1 的“DataSource”属性为“DataSource1”，设置 Query1 的“DatabaseName”属性，使其连接到数据库上。

（3）选中 TQuery 组件，在对象观察器中双击“SQL”属性打开字符串列表编辑器，如图 8.32 所示。



图 8.31　主窗体界面

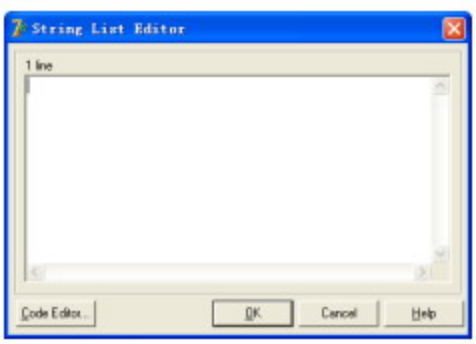


图 8.32　字符串列表编辑器

（4）在字符串列表编辑器中输入如下代码：

```
select * from 库存表
```

（5）单击 OK 按钮完成 SQL 语句输入。将“Active”属性设为“True”打开与数据表

的连接。

（6）设置 TRvDataSetConnection 组件的“DataSet”属性为“Query1”。如果 TQuery 组件所查询的数据表字段名称是中文，需要为中文字段名称起一个英文的别名，因为 Rave 报表不是非常支持中文字段，方法是在对象观察器中双击 TRvDataSetConnection 组件的“FieldAliasList”属性，在打开的字符串列表中为中文字段设置一个英文名称，如图 8.33 所示。

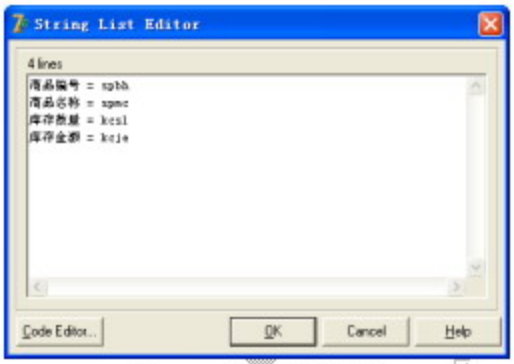


图 8.33　为字段设置别名

（7）单击 OK 按钮完成别名的设置。单击“Tools”/“Rave Designer”菜单项打开 Rave 报表工具，新建一个数据视图 DataView1。

（8）在报表中放置一个 TRegion 组件，调整 TRegion 组件的大小，在 TRegion 组件中放置 TBand 组件与 TDataBand 组件各一个。在 TBand 组件中放置 THLine 组件、TText 组件，选中一个 TText 组件，处理其“OnBeforePrint”事件，用于显示当前日期，具体步骤请参考 8.3.1 节步骤（6）～步骤（9）。

（9）设置 TDataBand 组件的“DataView”属性为“DataView1”，在 TDataBand 组件中放置 TDataText 组件、THLine 组件，设置 TDataText 组件的“DataView”属性与“DataField”属性，使其连接到指定的字段上。

（10）单击“File”/“Save”菜单项保存报表，将其保存在指定目录下“C:\Project5.rav”。

（11）在对象观察器中设置 TRvProject 组件的“ProjectFile”属性为报表文件，例如“C:\Project5.rav”。

（12）处理主窗体的 OnCreate 事件，代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ComboBox1.Clear;
  With Query1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select distinct 商品名称 From 库存表');
    Open;
  end;
  if Query1.RecordCount>0 then
    while Not Query1.Eof do
    begin
      ComboBox1.Items.Add(Query1.Fields[0].AsString);
      Query1.Next;
    end;
end;
```

（13）处理“查询”按钮的 OnClick 事件，代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
```

```
begin
  if Trim(ComboBox1.Text)<>'' then
  begin
    with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 库存表 where 商品名称 = :a');
      ParamByName('a').AsString := Trim(ComboBox1.Text);
      Open;
    end;
    if Query1.RecordCount>0 then
    begin
      BitBtn1.Enabled := True;
      DataSource1.DataSet := Query1;
    end
    else
    begin
      BitBtn1.Enabled := False;
      DataSource1.DataSet := Nil;
    end;
  end;
end;
```

（14）处理“打印”按钮的 OnClick 事件，代码如下：

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  RvProject1.Execute;
end;
```

（15）运行程序，单击“查询”按钮，主界面如图 8.34 所示。



图 8.34　程序主界面

（16）单击“打印”按钮，报表预览结果如图 8.35 所示。

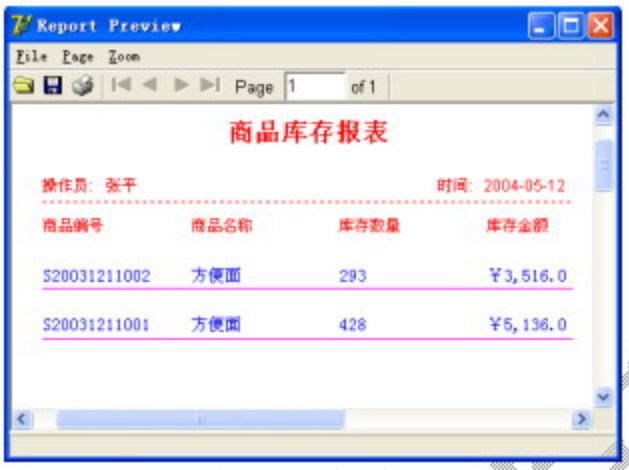


图 8.35　动态报表

8.4　利用向导创建报表

8.4.1　创建简单报表

1. 安装报表组件

在 Delphi 中提供了一组 TQReport 组件用来设计报表，但在 Delphi 7 中，该组件并没有安装到集成开发环境中，用户可以自己将 TQReport 组件安装到组件面板中，具体方法如下。

（1）单击“Component”/“Install Packages”菜单项，打开工程选项窗口，如图 8.36 所示。

（2）单击 Add 按钮打开添加包窗口，如图 8.37 所示。

（3）在 Delphi 7 的安装路径“Borland\Delphi7\Bin”下有一个“dclqrt70.bpl”文件，在添加包窗口中选中该文件，单击“打开”按钮返回到如图 8.36 所示界面，单击 OK 按钮完成报表组件的安装。

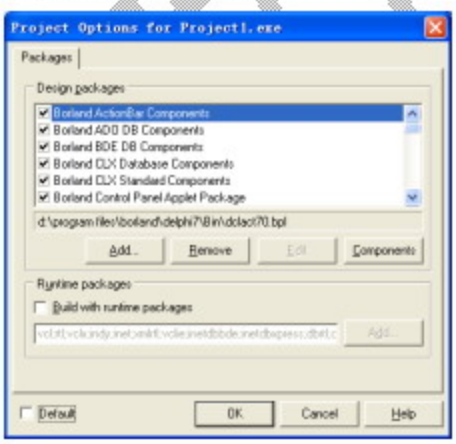


图 8.36　工程选项窗口

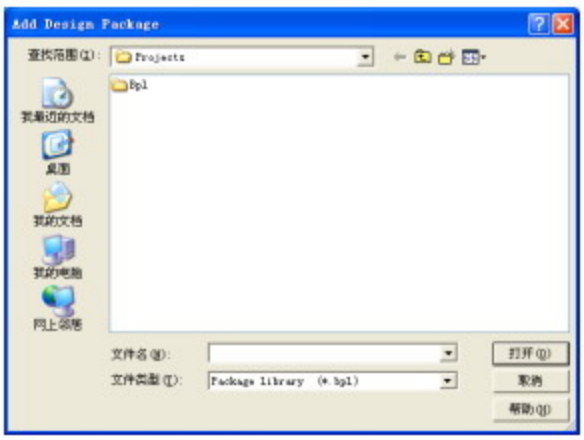


图 8.37　添加包窗口

2. 利用向导创建报表

（1）新建一个应用程序，在窗体中放置一个 TButton 组件。

（2）单击“File”/“New”/“Other”菜单项打开新建项目窗口，打开“Business”选项卡，如图 8.38 所示。

（3）选中“QuickReportWizard”选项，单击 OK 按钮打开新建报表向导窗口，如图 8.39 所示。

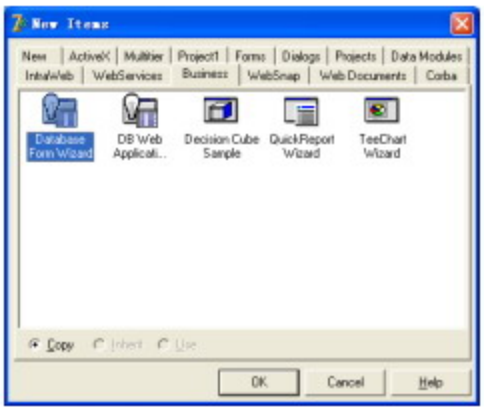


图 8.38　新建项目窗口

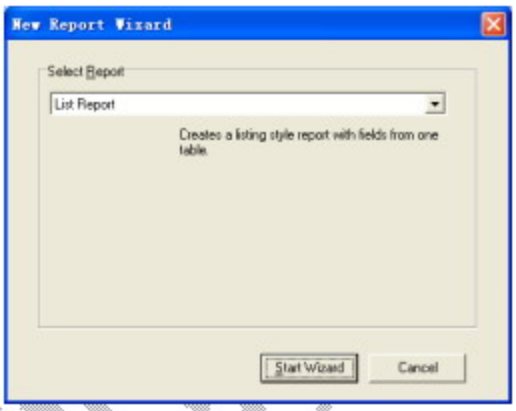


图 8.39　NewReportWizard 窗口

（4）单击“Start Wizard”按钮，切换至选择数据表页面，如图 8.40 所示。

（5）在“Alias or directory”下拉列表框中选择一个数据库别名，在“Table name”下拉列表框中选择一个数据表，如图 8.41 所示。

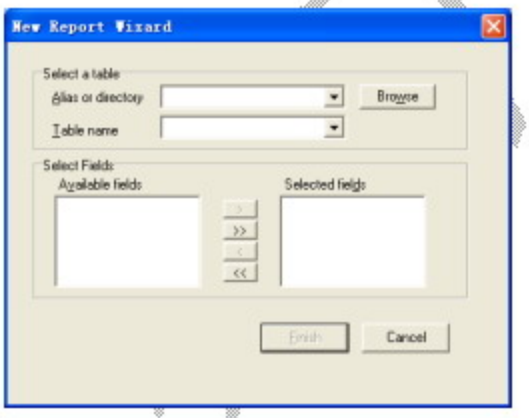


图 8.40　选择数据表 1

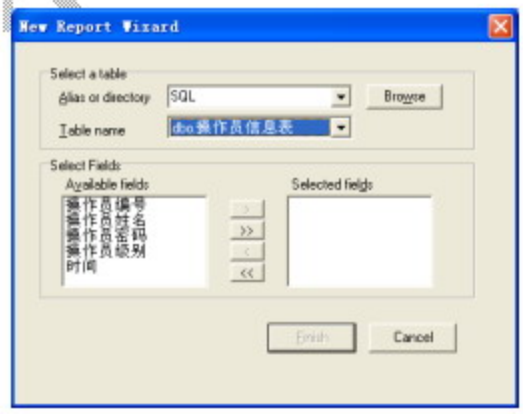


图 8.41　选择数据表 2

（6）单击“>>”按钮，选中所有字段，单击 Finish 按钮完成报表的创建。此时，在当前工程中会创建一个新的窗体，如图 8.42 所示。



图 8.42　报表窗体

（7）处理按钮的 OnClick 事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form2.QuickRep1.Preview;
end;
```

（8）运行程序，单击按钮，报表如图 8.43 所示。



图 8.43　报表预览结果

8.4.2　调整报表的外观

（1）打开如图 8.42 所示的窗体，选中 TQuickRep 组件，在对象观察器中展开“Bands”属性，设置“HasSummary”、“HasTitle”属性为“True”。界面如图 8.44 所示。



图 8.44　设计报表界面 1

（2）在标题边条 TitleBand1 中放置一个 TQRLabel 组件，设置该组件的“Caption”属性为“操作员报表”。在摘要边条 SummaryBand1 中放置两个 TQRLabel 组件，界面如图 8.45 所示。



图 8.45　设计报表界面 2

（3）处理报表创建的 OnCreate 事件，代码如下：

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  QRLabel8.Caption := FormatDateTime('yyyy-mm-dd',Now);
```

```
end;
```

（4）处理主窗体按钮的 OnClick 事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form2.QuickRep1.Preview;
end;
```

（5）运行程序，单击按钮，报表如图 8.46 所示。



图 8.46　报表预览结果

8.4.3　改变长度单位

在使用 TQuickRep 组件设计报表时，设计者可以使用几种不同的度量单位，TQuickRep 组件的“Units”属性提供了 4 种度量单位供用户选择，分别为：

- MM

表示毫米；

- Inches

表示英寸；

- Pixels

表示像素；

- Native

Native 是快速报表的内部单位，等于百分之一毫米。

当用户选择不同的度量单位时，报表会自动地进行数据转换。

8.5　使用 QuickReport 组件设计报表

8.5.1　图案报表设计

图案报表是将数据表中某些图形数据（通常是二进制数据）显示在报表中，具体设计步骤如下。

（1）新建一个应用程序，在窗体上放置 TQuickRep、TQRLabel、TQRDBText、TButton、TQRDBImage、TQRShape、TTable 等组件。在对象观察器中展开 TQuickRep 组件的“Bands”

属性，设置“HasColumnHeader”、“HasDetail”、“HasTitle”属性为“True”，界面如图 8.47 所示。



图 8.47　图形报表界面

（2）设置 TTable 组件的“DatabaseName”、“TableName”属性，使其连接到数据表中，将“Active”属性设为“True”，打开与数据表的连接。

（3）设置 TQuickRep 组件的“DataSet”属性为“Table1”，设置 TQRDBText 组件与 TQRDBImage 组件的“DataSet”、“DataField”属性，使其连接到数据表中指定的字段。

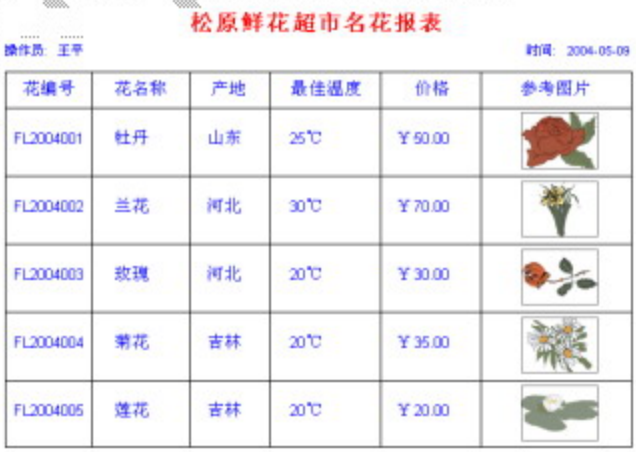
（4）处理窗体的 OnShow 事件，代码如下：

```
procedure TForm1.FormShow(Sender: TObject);
begin
  QRLabel11.Caption := FormatDateTime('yyyy-mm-dd',Now);
end;
```

（5）处理按钮的 OnClick 事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  QuickRep1.Preview;
end;
```

（6）运行程序，单击按钮，报表预览结果如图 8.48 所示。


松原鲜花超市名花报表

操作员: 王平　　时间: 2004-05-09

花编号	花名称	产地	最佳温度	价格	参考图片
FL2004001	牡丹	山东	25℃	￥50.00	
FL2004002	兰花	河北	30℃	￥70.00	
FL2004003	玫瑰	河北	20℃	￥30.00	
FL2004004	菊花	吉林	20℃	￥35.00	
FL2004005	莲花	吉林	20℃	￥20.00	

图 8.48　图形报表

8.5.2　设计连续打印报表

（1）新建一个应用程序，在窗体中放置 TQuickRep、TTable、TQRLabel、TQRDBText、TButton、

TLabel、TEdit、TQRShape组件。在对象观察器中展开TQuickRep组件的“Bands”属性，设置“HasColumnHeader”、“HasDetail”、“HasTitle”属性为“True”，界面如图8.49所示。



图8.49 连续打印报表界面

（2）设置TTable组件的“DatabaseName”、“TableName”属性，使其连接到数据表中，将“Active”属性设置为“True”，打开与数据表的连接。

（3）设置TQuickRep组件的“DataSet”属性为“Table1”，设置TQRDBText组件的“DataSet”、“DataField”属性，使其连接到指定的字段上。

（4）处理“打印”按钮的OnClick事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  QuickRep1.PrinterSettings.Copies := StrToInt(Edit1.Text);
  QuickRep1.Print;
end;
```

（5）处理“预览”按钮的OnClick事件，代码如下：

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  QuickRep1.Preview;
end;
```

（6）处理TEdit组件的OnExit事件，代码如下：

```
procedure TForm1.Edit1Exit(Sender: TObject);
begin
  if (Trim(Edit1.Text)='')or(Trim(Edit1.Text)='0') then
    Edit1.Text := '1';
end;
```

（7）处理TEdit组件的OnKeyPress事件，代码如下：

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var
  m: Boolean;
begin
  m := (Key<#8)or(Key>#8)and(Key<#48)or(Key>#57);
  if m then
    Key := #0;
```

```
end;
```

（8）处理窗体的 OnShow事件，代码如下：

```
procedure TForm1.FormShow(Sender: TObject);
begin
  QRLabel9.Caption := FormatDateTime('yyyy-mm-dd',Now);
end;
```

（9）运行程序，在编辑框中输入报表需要打印的份数，单击“打印”按钮，打印机会连续打印出指定数量的报表。报表效果如图 8.50 所示。



商品基础信息报表

操作员： 王南　　时间： 2004-05-09

商品编号	商品名称	简称	规格	进价	批发价
S20031211001	方便面	FBM	30袋/箱	￥12.00	￥22.00
S20031211002	方便面	FBM	20袋/箱	￥12.00	￥18.00
S20031211003	苹果	PG	30个/箱	￥22.40	￥36.00
S20031211004	香蕉	BN	30斤/箱	￥30.00	￥38.00

图 8.50　连续打印的报表

8.6　数据报表应用总结

8.6.1　设计报表版面

一个完整的报表一般包括页眉、报表标题、正文、摘要、页脚 5部分，其中正文又分为列标题与数据两部分。在设计报表时，报表标题、正文是不可缺少的部分，如果报表中有页眉、摘要、页脚，页眉应放在报表标题的上方，摘要在正文的下方，页脚放在报表的最底部。

8.6.2　报表设计过程

（1）确定数据源，即报表中正文需要显示的数据。
（2）确定报表框架，即报表由几部分组成。
（3）使用数据集组件从数据库中读取数据源。
（4）使用报表组件布置报表界面。
（5）设置报表组件相关属性使其连接到数据集组件上。
（6）使用按钮或其他组件设置报表的预览及打印功能。
（7）运行程序，检查报表是否实现了预期的功能。

8.6.3　报表设计要点

- 报表首先需要实现客户需求，即以实现预期功能为首要目标。
- 报表版面要美观、得体。
- 报表打印功能要灵活，能够实现各种复杂报表的打印。
- 操作简单，让用户容易掌握报表的各种操作。

调试程序与预错处理

第9章

9.1 错误的种类

编写健壮的代码是程序员需要具备的能力，开发高品质的软件是程序员的追求。然而，在实际开发过程中，错误的产生是不可避免的，由于程序员编写代码的疏忽、软件设计框架的缺陷以及硬件环境的影响都会导致错误的产生。在 Delphi 中，错误的种类多种多样，从错误产生的时机来说，主要有两类：设计时错误与运行时错误。

1. 设计时错误

设计时错误是指软件在编译时产生的错误，通常是代码的语法错误。这类错误比较容易发现和处理。

2. 运行时错误

运行时错误是指操作员在操作软件的过程中产生的错误。这类错误产生的原因很多，例如，数据库访问错误、数组越界、操作员误操作等，这类错误比较隐蔽，难于发现和处理。

9.2 错误捕获

9.2.1 错误捕捉

1. 系统自动捕捉错误

程序在执行过程中产生错误，如果不进行处理，程序很可能会中断离开，无法再回到错误产生前的状况继续执行程序。

在 Delphi 的开发环境中，如果程序在运行时产生错误，即使用户不进行任何错误处理，Delphi 也会捕捉到这个错误，并以对话框的形式显示异常信息，然后让程序继续执行。

2. 用户捕捉错误

虽然 Delphi 能够自动捕捉错误并进行处理，但只是进行一般的处理，即避开发生异常的程序代码。它没有排除掉错误发生的原因，如果保持原来的状态再进行相同的操作，仍然会发生同样的错误。为了让程序能够更顺畅地执行，开发人员需要在程序中对可能产生的错误进行处理，当错误产生时，使程序能够按预先目的对错误进行特殊处理。

9.2.2 触发错误

在 Delphi 中，当程序产生错误时，系统会自动触发错误，但是开发人员也可以根据需要自行触发错误。在程序中，可以使用 Raise 指令触发错误，其语法如下：

```
Raise  异常类的实例
```

说明：Raise 语句应放在错误处理语句（错误处理语句会在下一节中介绍）中，否则，Raise 语句之后的程序不会执行，因为该语句触发了错误，程序产生中断，但是用户并没有进行处理，而由系统进行默认的处理。

示例如下：

```
Edit1.Clear;
Raise  Exception.Create('触发错误');
Edit2.Clear; //不会执行
Edit3.Clear; //不会执行
```

在上面的代码中，后面的两句代码永远也不会执行，但是如果同样的代码出现在错误处理语句中，情况就不同了。示例如下：

```
Edit1.Clear;
Try
Raise  Exception.Create('触发错误');
Except
Edit2.Clear; //错误产生时执行
End;
Edit3.Clear; //正常执行
```

在上面的代码中，语句“Edit2.Clear”在错误产生时执行，而语句“Edit3.Clear”在错误产生时也能够执行。

9.2.3 编制错误处理程序

在 Delphi 中，提供了专门用于异常处理的语句，使程序员能够对程序中产生的异常进行处理，防止程序中断。

1. Try…Except…End

使用该语句可以让开发人员自行捕捉异常，然后根据异常的不同类型做不同的处理。请看下面的语法内容：

```
Try
```

```
    语句；//可能产生异常的语句
    ...
  Except
    语句；//处理异常的语句
    ...
  End;
```

如果“Try…Except”之间的语句没有产生异常，程序会跳过“Except…End”区内的程序代码，执行其后面的程序，如果“Try…Except”之间的语句产生异常，程序就会从产生异常的位置跳到“Except…End”区，执行“Except…End”之间的程序代码。示例如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Try
    StrToInt(Edit1.Text);
  Except
    ShowMessage('请输入合法整数.');
    Edit1.Clear;
  End;
end;
```

2. Try…Finally…End

“Try…Finally”之间是可能发生异常的语句，“Finally…End”之间的程序代码无论是否有异常产生，都将会执行。示例如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Try
    StrToInt(Edit1.Text);
  Finally
    ShowMessage('请输入合法整数.');
    Edit1.Clear
  End;
end;
```

请读者注意上面两个示例的区别，在第一个示例中，只有在出现异常时才出现对话框，而在第二个示例中，无论是否出现异常，都会出现提示对话框。

9.3 代码调试

9.3.1 跟踪程序的执行

在开发程序过程中，经常会出现各种错误，作为一名程序开发人员，有责任对错误进行查找和处理。在 Delphi 中提供了单步跟踪功能，使程序员可以单行观察程序执行情况。例如，在单击一个按钮时出现异常，程序员可以对按钮的单击事件进行跟踪，方法是在按钮的单击事件代码中设置断点，即在按钮的单击事件代码中按 F5 键，代码编辑器中当前行会变为红色，表示在当前行设置了断点。效果如图 9.1 所示。

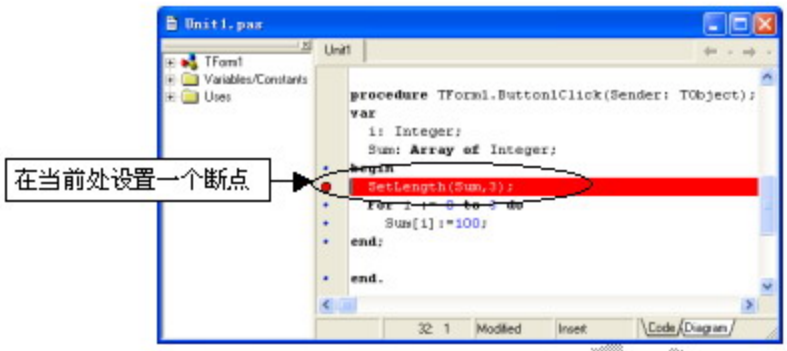


图 9.1　设置断点

运行程序，单击按钮，程序会停留在断点处，如图 9.2 所示。

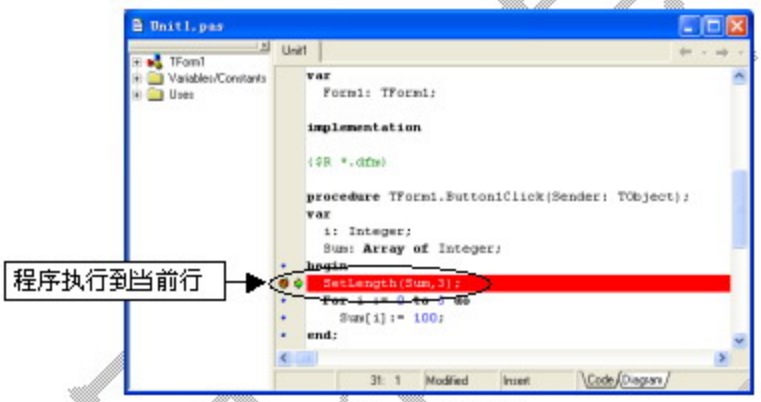


图 9.2　跟踪程序

如果程序员按 F7 键，程序会一行一行执行代码。需要注意的是，当绿色箭头出现在当前行时，表示程序停留在当前处，当前行代码此时还没有被执行。

9.3.2　Call Stack 窗口

在 Delphi 的开发环境中提供了堆栈窗口，用于查看应用程序调用的函数和过程。单击“View”/“Debug Windows”/“Call Stack”菜单项可以打开调用栈窗口，如图 9.3 所示。

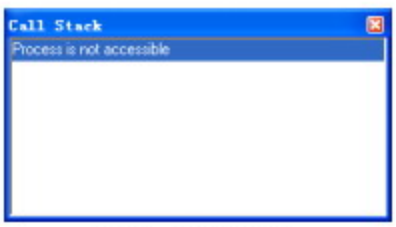


图 9.3　调用栈窗口

如果程序处于调试阶段，调用该窗口会显示程序当前调用的过程或函数信息。例如，在按钮的单击事件中调用一个过程，代码如下：

```
Procedure SetCaption(Text : String);
begin
   Form1.Caption := Text;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
   SetCaption('Call Stack');
end;
```

在按钮的单击事件中设置一个断点，运行程序，单击按钮，程序会停留在断点处，如图 9.4 所示。

按 F7 键执行当前语句，程序会跳转到“SetCaption”过程中，打开调用栈窗口，在调用栈窗口中会显示程序当前调用的过程和函数信息，如图 9.5 所示。

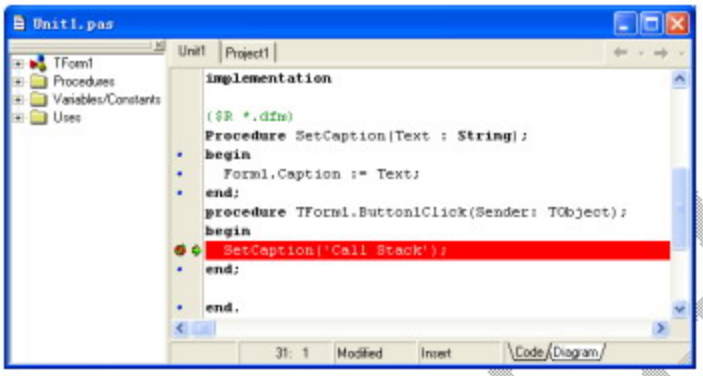


图 9.4　代码编辑器窗口

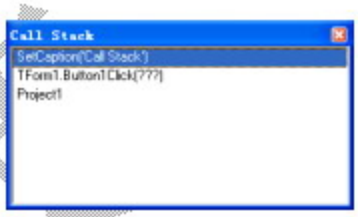


图 9.5　调用栈窗口信息

9.3.3　Watch List 窗口

在 Delphi 中，为了方便开发人员在程序运行时查看变量值，提供了 Watch List 窗口。用户可以单击“View”/“Debug Windows”/“Watches”菜单项打开该窗口，也可以直接按 Ctrl+Alt+W 组合键打开 Watch List 窗口，如图 9.6 所示。

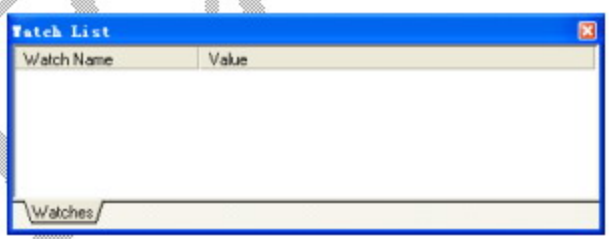


图 9.6　Watch List 窗口 1

下面将以一个具体示例来介绍 Watch List 窗口的使用方法。

（1）处理一个按钮的单击事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.Text := '10';
  Edit2.Text := '11';
  Edit3.Text := '12';
  Caption := '14';
end;
```

（2）在代码“Edit3.Text := '12'”处设置一个断点，运行程序，单击按钮，使程序停留在断点处，如图 9.7 所示。

（3）按 Ctrl+Alt+W 组合键打开 Watch List 窗口，在 Watch List 窗口中用鼠标右键单击会弹出一个快捷菜单，如图 9.8 所示。

（4）如果在“Expression”下拉列表框中输入“Edit3.Text”，会发现其值并不为“12”，

如图 9.9 所示。

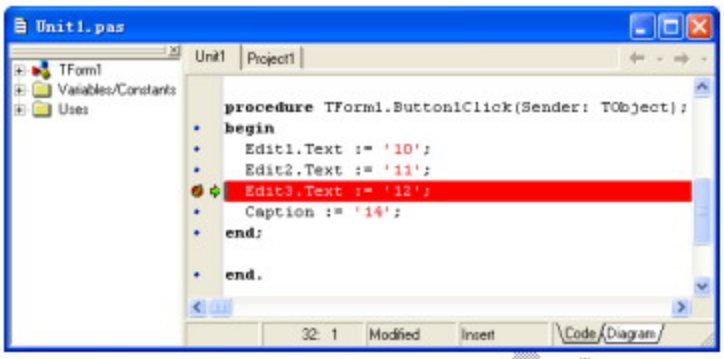


图 9.7　代码编辑器 1

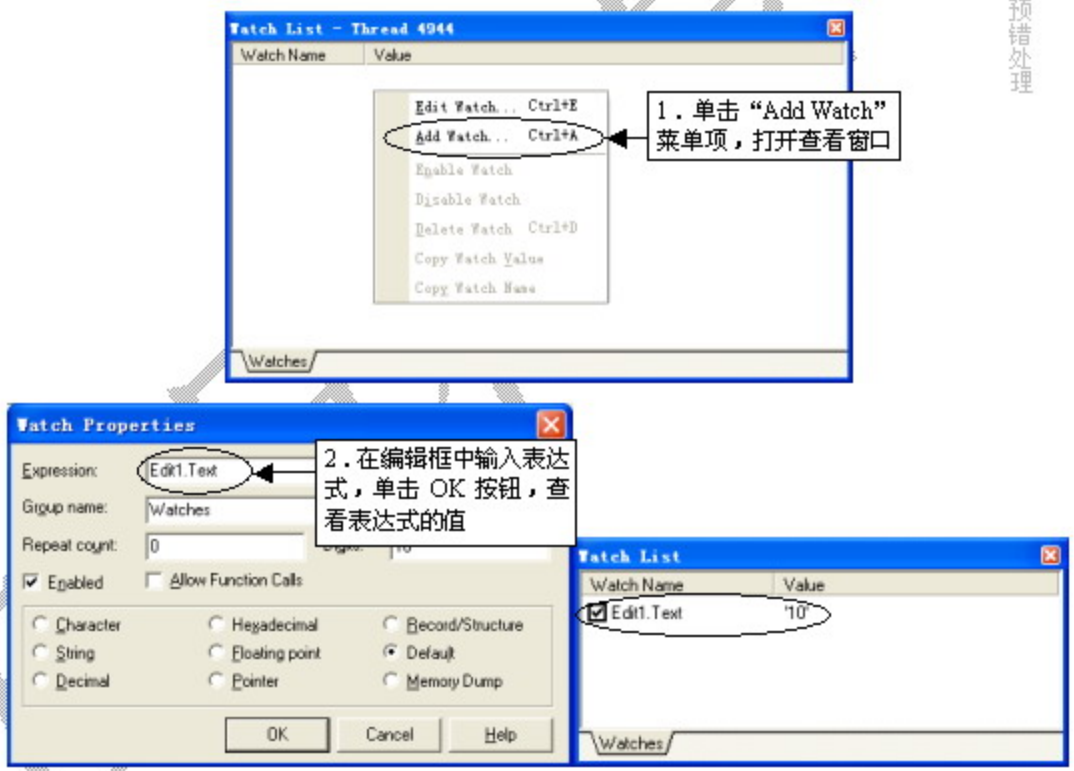


图 9.8　查看表达式

因为当前“Edit3.Text := '12'”语句还没有执行，如果按 F7 键执行当前代码，会发现“Edit3.Text”值为“12”，如图 9.10 所示。

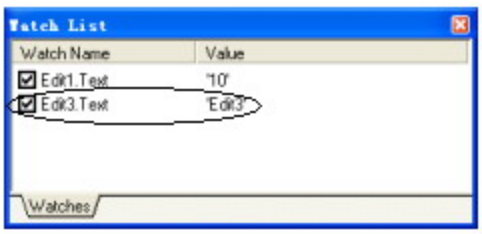


图 9.9　Watch List 窗口 2

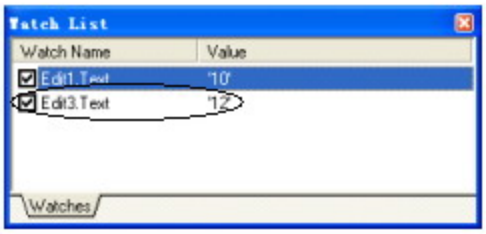


图 9.10　Watch List 窗口 3

9.3.4 Debug Inspector 窗口

使用 Debug Inspector 窗口可以查看类、对象、属性、记录等信息。用户可以单击“Run”/“Inspect”菜单项打开 Inspect 窗口，在 Inspect 窗口中输入类、对象等信息单击 OK 按钮便可以打开 Debug Inspector 窗口。需要说明的是，只有在程序处于调试阶段（程序暂停执行）时 Debug Inspector 窗口才可用。

（1）处理按钮的单击事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.Clear;
  Edit2.Clear;
  Edit3.Clear;
  Caption := '14';
end;
```

（2）在代码“Edit2.Clear”处设置一个断点，运行程序，单击按钮，使程序停留在断点处，如图 9.11 所示。

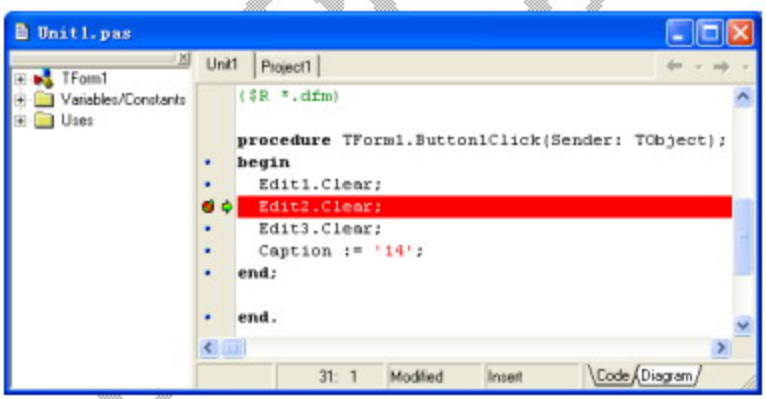
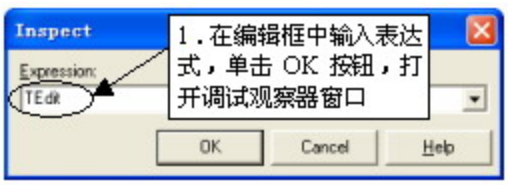


图 9.11 代码编辑器 2

（3）单击“Run”/“Inspect”菜单项或者直接按 Alt+F5 组合键打开 Inspect 窗口，如图 9.12 所示。



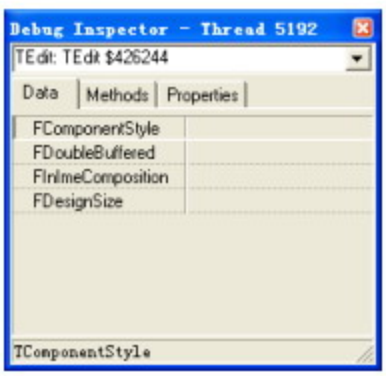


图 9.12 查看 TEdit 类

在 Debug Inspector 窗口中显示了 TEdit 类的所有属性和方法，如果用户在 Inspect 窗口

中输入属性，例如“Edit1.Text”，单击 OK 按钮打开 Debug Inspector 窗口，在 Debug Inspector 窗口中会显示其属性值，如图 9.13 所示。

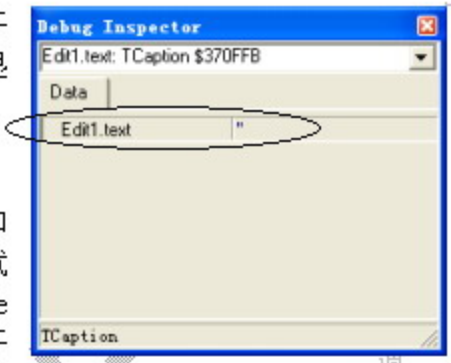


图 9.13　Debug Inspector 窗口

9.3.5　Evaluate/Modify 窗口

利用 Evaluate/Modify 窗口可以在调试阶段观察和修改变量的值，使用它可以修改某个变量的值以测试不同的输出结果。用户可以单击“Run”/“Evaluate Modify”菜单项或者直接按 Ctrl+F7 组合键打开 Evaluate/Modify 窗口，如图 9.14 所示。

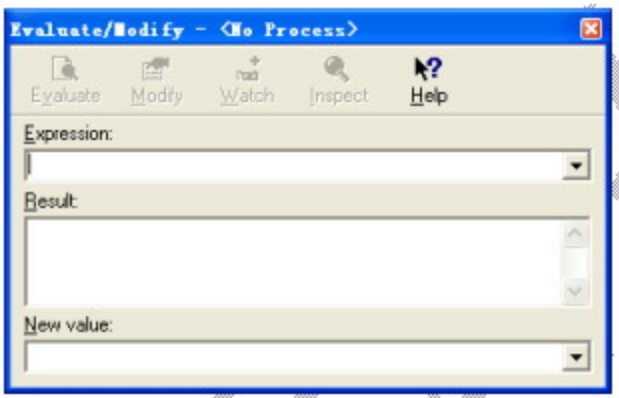


图 9.14　Evaluate Modify 窗口 1

在 Evaluate/Modify 窗口中会发现有几个选项不可用，这些选项只有在程序处于调试阶段时才可用。下面结合具体示例来介绍 Evaluate/Modify 窗口的使用。

（1）处理按钮的单击事件，代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
var
   i: String;
begin
   i:= 'aaa' ;
   Edit1.Text := i;
   Edit2.Text := i;
   Edit3.Text := i;
end;
```

（2）在代码“Edit1.Text := i”处设置断点，运行程序，单击按钮使程序停留在断点处，效果如图 9.15 所示。

（3）打开 Evaluate/Modify 窗口，在“Expression”下拉列表框中输入要查看或修改的变量，例如“i”，按 Enter 键，在 Evaluate/Modify 窗口中会显示变量“i”的当前值，如图 9.16 所示。

（4）在“New value”下拉列表中输入一个新值，例如“Modify”，单击 Modify 按钮更改变量“i”的值，如图 9.17 所示。

（5）按 F9 键运行程序，会发现编辑框中的文本为“Modify”。

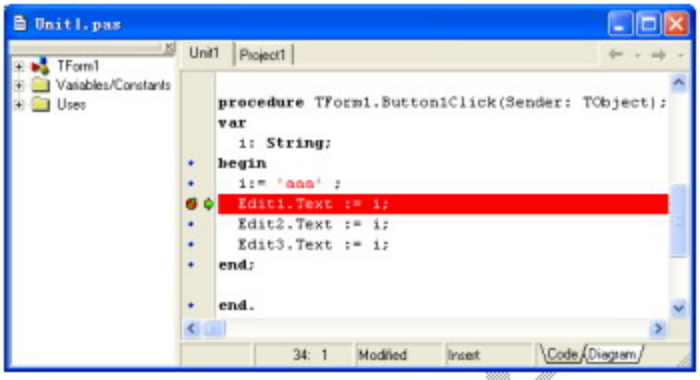


图 9.15　代码编辑器 3



图 9.16　Evaluate Modify 窗口 2

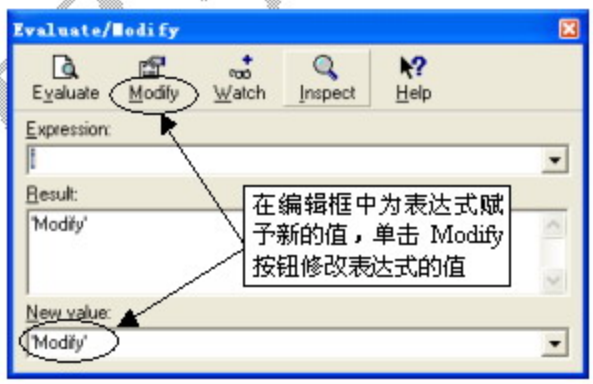


图 9.17　Evaluate Modify 窗口 3

工欲善其事 必先利其器

2

第二篇
SQL Server 应用篇

- 安装SQL Server 2000
- 配置SQL Server服务器与客户端
- 数据备份与恢复
- 数据库附加与分离
- 导入、导出数据表
- 设计数据库
- 修改数据库
- 创建关系与视图
- 录入和修改数据
- 浏览和查询数据
- SQL语句
- 应用存储过程
- 应用触发器
- 应用视图

第10章 管理 SQL Server 2000

第11章 设计 SQL Server 数据库

第12章 编辑 SQL Server 数据库

第13章 SQL 语句

第14章 存储过程、触发器与视图

管理 SQL Server 2000

第 10 章

10.1 安装 SQL Server

（1）将 SQL Server 2000 安装盘放入光驱，光盘会自动运行，弹出如图 10.1 所示界面。



图 10.1　启动界面

（2）在启动界面中选择“安装 SQL Server 2000 组件”选项，切换至“安装组件”界面，如图 10.2 所示。

（3）在“安装组件”界面中选择“安装数据库服务器”选项，打开 SQL Server 安装向导，如图 10.3 所示。

（4）单击“下一步”按钮，打开“计算机名”对话框，如图 10.4 所示。



图 10.2　安装组件



图 10.3　SQL Server 安装向导



图 10.4　计算机名

（5）在“计算机名”对话框中选择安装 SQL Server 数据库实例的计算机，这里选择“本地计算机”选项。如果用户想要在网络上其他计算机中安装 SQL Server 数据库实例，可以选中“远程计算机”选项，然后输入计算机名称，也可以单击“浏览”按钮，在弹出的“选择计算机”对话框中选择计算机。单击“下一步”按钮，打开“安装选择”对话框，如图 10.5 所示。

（6）在“安装选择”对话框中选择安装方式，这里选择“创建新的 SQL Server 实例，或安装客户端工具”选项，单击“下一步”按钮打开“用户信息”对话框，如图 10.6 所示。

（7）在“姓名”和“公司”编辑框中输入用户姓名和公司名称，这些信息也可以省略。单击“下一步”按钮打开“软件许可证协议”对话框，如图 10.7 所示。

（8）单击“是”按钮进入“安装定义”对话框，如图 10.8 所示。

（9）在“安装定义”对话框中有 3 个安装类型，“仅客户端工具”选项表示只安装客户端数据库管理工具。“服务器和客户端工具”选项表示安装服务器和客户端工具以创建具有管理能力的关系数据库服务器。选择“仅连接”选项，表示只安装数据库客户端连接组件。这里选择“服务器和客户端工具”选项，单击“下一步”按钮，打开“实例名”对话框，如图 10.9 所示。

（10）在“实例名”对话框中设置安装 SQL Server 的实例名称，可以使用默认的实例名，即选中“默认”复选框。如果要自己设置实例名，不选中“默认”复选框，在“实例名”编辑框中输入实例名，这里选择默认的实例名。单击“下一步”按钮，打开“安装类

型”对话框，如图 10.10 所示。

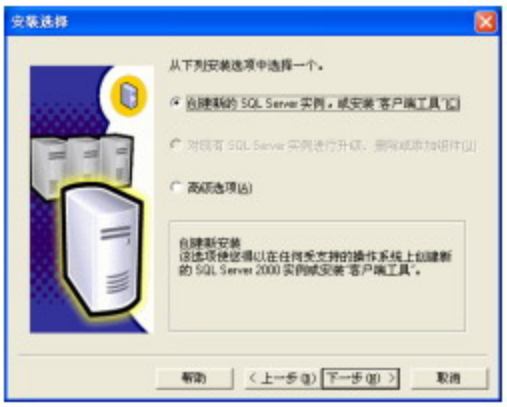


图 10.5　安装选择

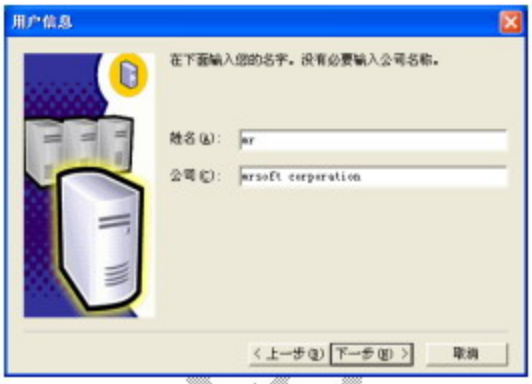


图 10.6　用户信息

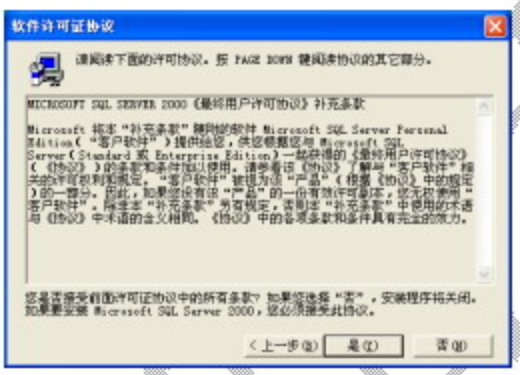


图 10.7　软件许可证协议

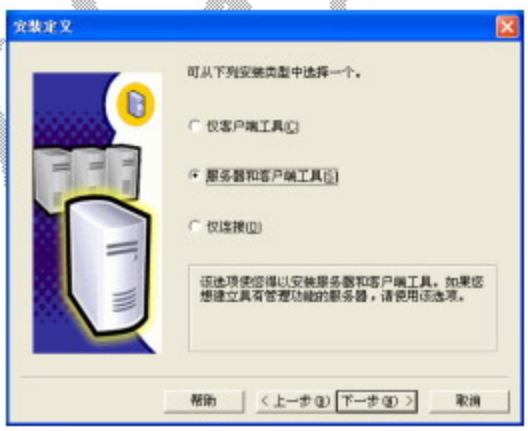


图 10.8　安装定义

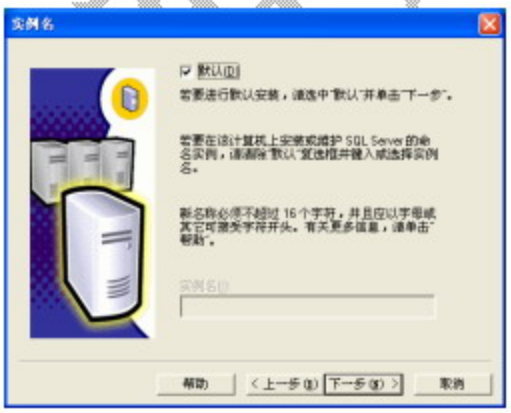


图 10.9　实例名

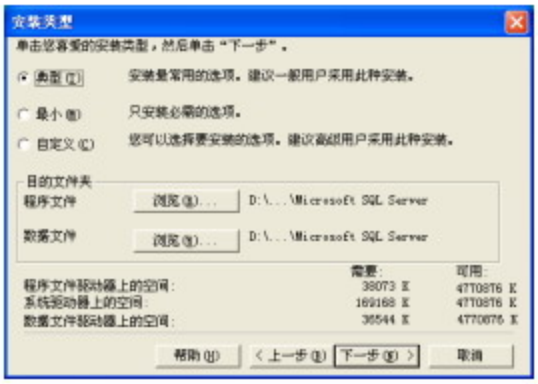


图 10.10　安装类型

（11）在“安装类型”对话框中选择“典型”选项，即进行典型安装，如果用户要更改默认的安装路径，可以单击“浏览”按钮重新指定安装路径。这里选择默认设置，单击“下一步”按钮，打开“服务账户”对话框，如图 10.11 所示。

（12）在“服务账户”对话框中设置用户账户，在“服务设置”选项组中选择“使用本地系统账户”选项，单击“下一步”按钮，打开“身份验证模式”对话框，如图 10.12 所示。

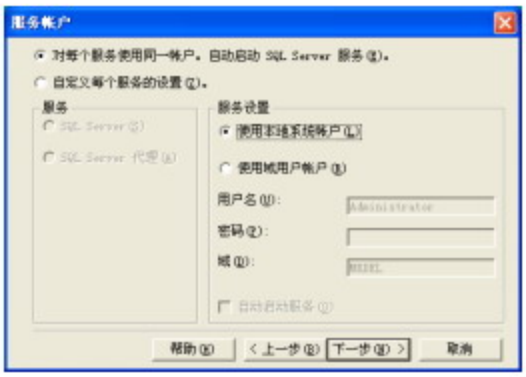


图 10.11　服务账户

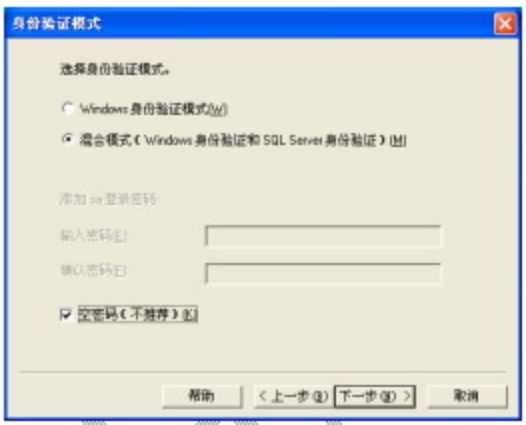


图 10.12　身份验证模式

（13）在“身份验证模式”对话框中选择“混合模式”选项，同时选中“空密码”复选框，单击“下一步”按钮，打开“开始复制文件”对话框，如图 10.13 所示。

（14）在“开始复制文件”对话框中单击“下一步”按钮开始复制文件，即对 SQL Server 进行安装，安装成功后，将弹出“安装完毕”对话框，如图 10.14 所示。

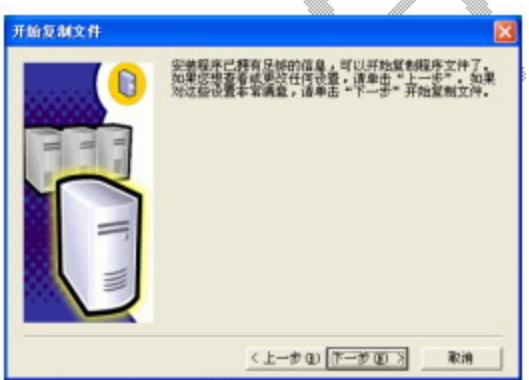


图 10.13　开始复制文件



图 10.14　安装完毕

10.2　配置服务器

10.2.1　启动、暂停和停止 SQL Server 服务器

在安装完 SQL Server 后，需要启动 SQL Server 服务器才能使用数据库。在“开始”菜单中单击“服务管理器”菜单项，打开 SQL Server 服务管理器，如图 10.15 所示。

单击“开始/继续”按钮启动服务器，如图 10.16 所示。

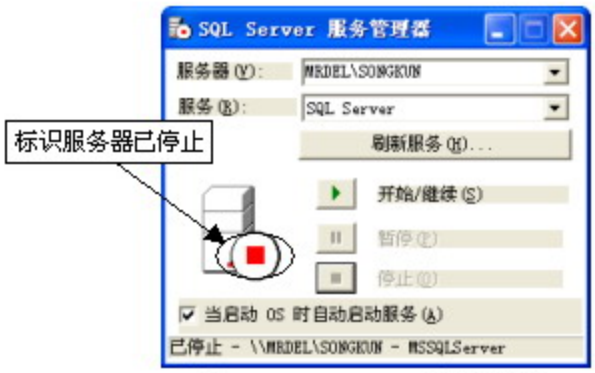


图 10.15　SQL Server 服务管理器

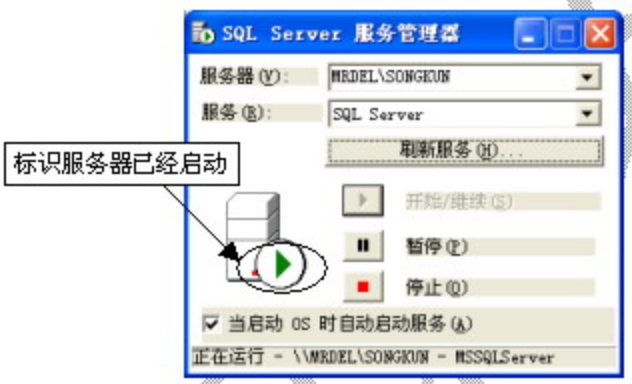


图 10.16　启动服务器

在 SQL Server 服务器首次启动后，图标会出现在任务栏中，如图 10.17 所示。用户在需要打开“SQL Server 服务管理器”时，只要单击任务栏上的图标即可。



图 10.17　任务栏中的 SQL Server 服务管理器图标

通过 SQL Server 服务管理器，用户可以启动、暂停、停止 SQL Server 服务器，用户只要单击相应的按钮即可，但通常情况下，不要暂停或停止 SQL Server 服务器，否则数据库将无法使用。

10.2.2　注册 SQL Server 2000 服务器

默认情况下，在安装完 SQL Server 2000 后，系统会自动将本机中的实例注册到 SQL Server 服务器中。但是，如果用户想要访问网络上的其他 SQL Server 服务器，就需要注册目标服务器了。

（1）在“开始”菜单中单击“企业管理器”菜单项，打开企业管理器，如图 10.18 所示。

（2）用鼠标右键单击“SQL Server 组”节点，在弹出的快捷菜单中选择“新建 SQL Server 注册”项，打开“注册 SQL Server 向导”，如图 10.19 所示。

（3）单击“下一步”按钮，进入选择服务器界面，如图 10.20 所示。

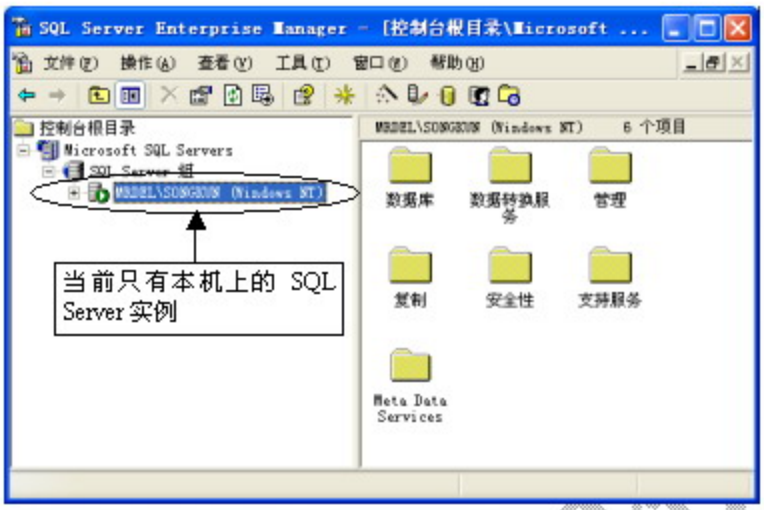


图 10.18　企业管理器



图 10.19　注册 SQL Server 向导



图 10.20　选择服务器

（4）在“可用的服务器”列表中选中需要注册的服务器，单击“添加”按钮，将其添加到“添加的服务器”列表中，此时“下一步”按钮可用，单击该按钮，进入“选择身份验证模式”界面，如图 10.21 所示。



图 10.21　选择身份验证模式

（5）选中“系统管理员给我分配的 SQL Server 登录信息（SQL Server 身份验证）”选项，单击“下一步”按钮，进入“选择连接选项”界面，如图 10.22 所示。

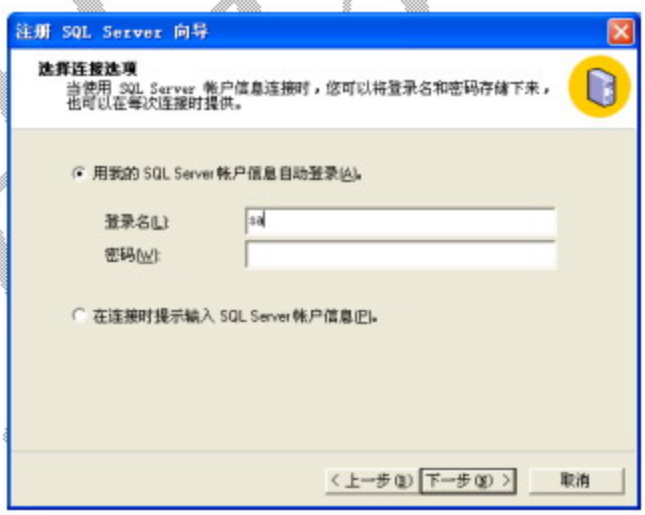


图 10.22　选择连接选项

（6）选中“用我的 SQL Server 账户信息自动登录”选项，在“登录名”编辑框中输入“sa”（超级用户），在“密码”编辑框中输入密码，如果密码为空，可以不输入。单击“下一步”按钮，进入“选择 SQL Server 组”界面，如图 10.23 所示。

（7）保持默认设置，单击“下一步”按钮，进入“完成注册”界面，单击“完成”按钮，注册服务器，如图 10.24、图 10.25 所示。



图 10.23　选择 SQL Server 组



图 10.24　完成注册



图 10.25　服务器注册状态

（8）当服务器注册完成后，企业管理器中将显示该服务器，如图 10.26 所示。

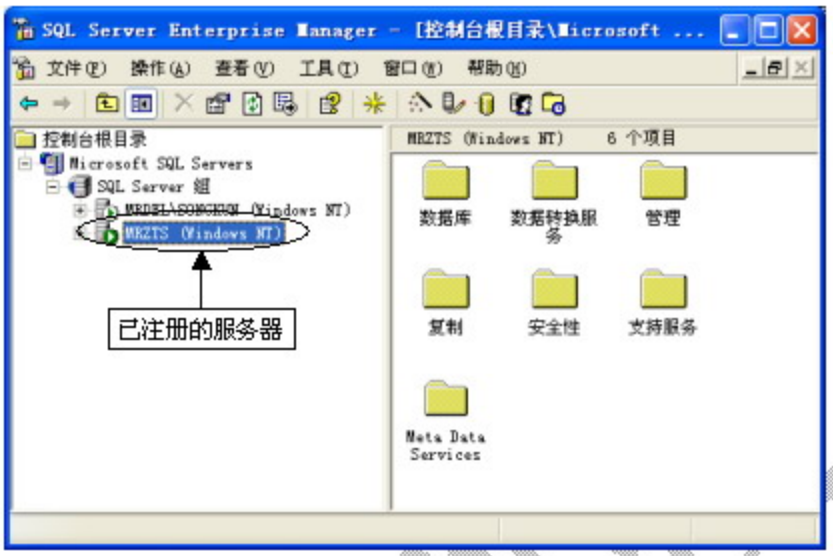


图 10.26 企业管理器窗口

10.2.3 编辑 SQL Server 2000 注册信息

当用户注册了一个网络服务器，还可以通过企业管理器修改注册服务器的信息。例如，更改与网络服务器的身份验证方式。

（1）在企业管理器中选择一个已注册的服务器，例如在 10.2.2 节注册的“MRZTS”，上右击，在弹出的快捷菜单中选择“编辑 SQL Server 注册属性”菜单项，打开“已注册的 SQL Server 属性”对话框，如图 10.27 所示。

（2）在“已注册的 SQL Server 属性”对话框中可以修改服务器的身份验证方式。例如设置“使用 Windows 身份验证”或“使用 SQL Server 身份验证”验证方式，单击“确定”按钮保存设置。

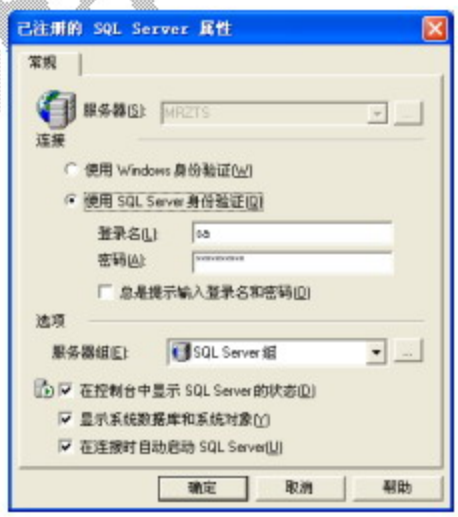


图 10.27 已注册的 SQL Server 属性

10.3 脚本概述

10.3.1 将数据库生成脚本

（1）打开企业管理器，展开“数据库”选项，选中欲生成脚本的数据库，如图 10.28 所示。

（2）切换至“选项”选项卡，为 SQL 脚本设置选项，生成脚本，具体操作如图 10.29 所示。

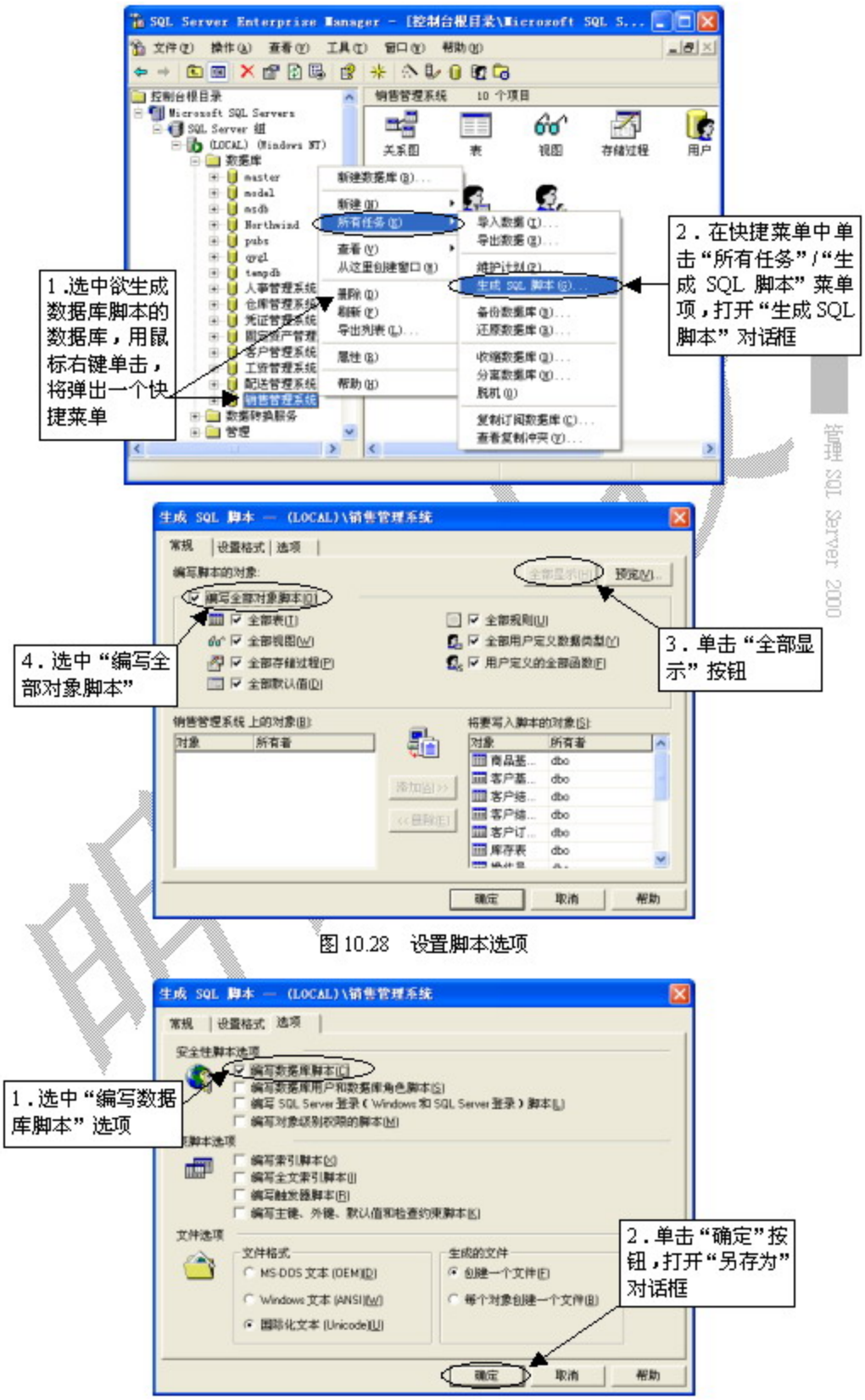


图 10.28　设置脚本选项



图 10.29　编写数据库脚本

10.3.2　将指定表生成脚本

生成表脚本与生成数据库脚本步骤大致相同，只是在步骤（2）中不选中“编写数据库脚本”检查框，具体步骤请参考 10.3.1 小节。表脚本与数据库脚本的区别在于表脚本会在当前数据库中创建数据表、视图、存储过程等数据库对象，而数据库脚本会创建一个新的数据库，并在该数据库中创建各个数据库对象。

10.3.3　执行脚本

脚本的执行通常是在查询分析器中进行的。在“开始”菜单中打开查询分析器，如图 10.30 所示。

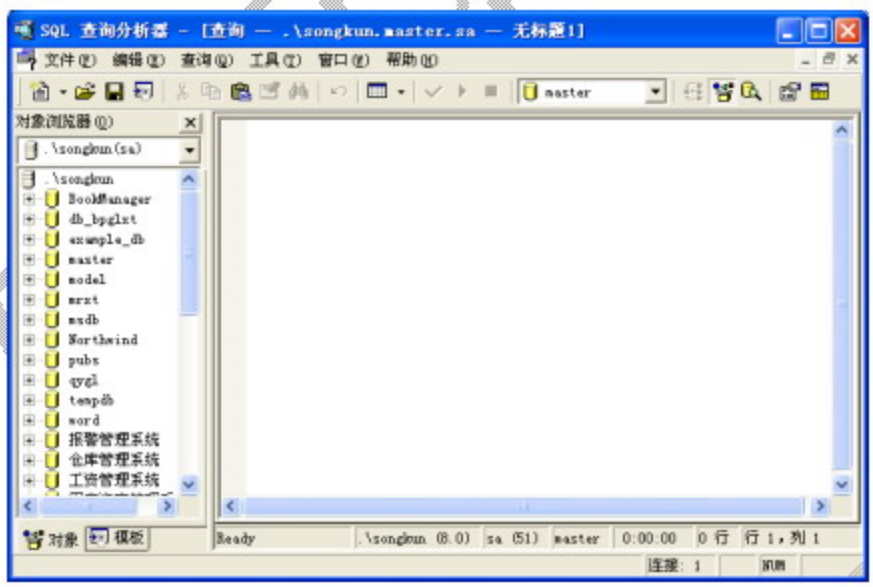


图 10.30　查询分析器

在查询分析器中单击“文件”/“打开”菜单项，打开“打开查询文件”对话框，如图 10.31 所示。

在“打开查询文件”对话框中选择需要执行的脚本，如“wage.sql”，单击“打开”按钮打开脚本，如图 10.32 所示。

在查询分析器中按 F5 键执行脚本中的 SQL 语句。

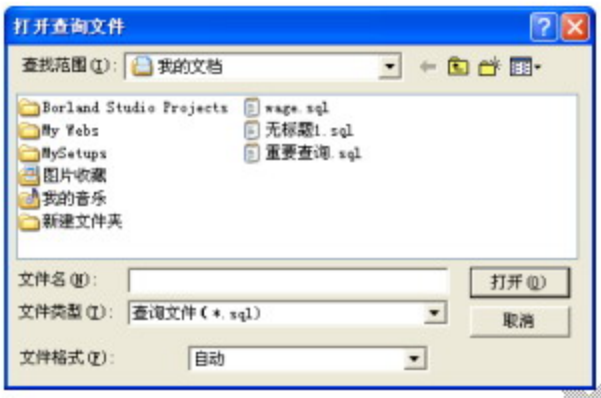


图 10.31　打开查询文件窗口

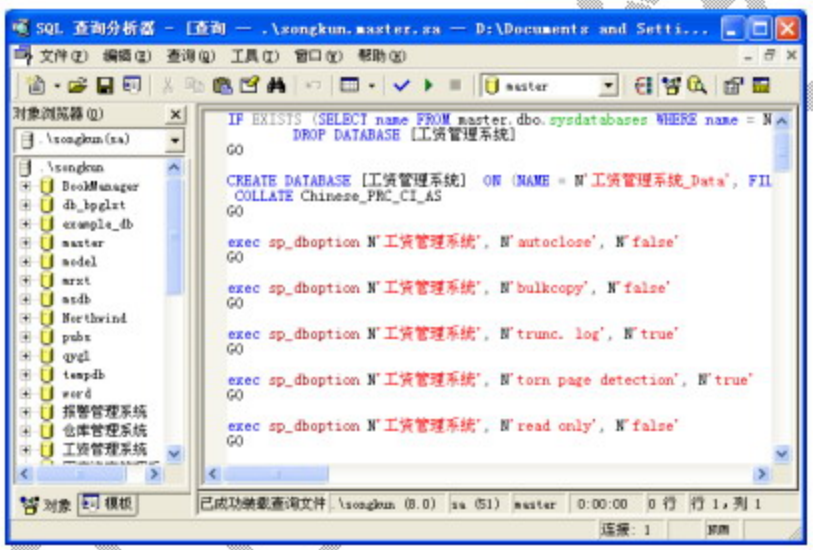


图 10.32　加载数据库脚本

10.4　备份和还原数据库

10.4.1　备份和恢复的概念

备份数据库是指对数据库或事务日志进行复制，当系统、磁盘或数据库文件损坏时，可以使用备份文件进行恢复，防止数据丢失。

还原数据库是使用数据库的备份文件对数据库进行还原操作。由于病毒的破坏、磁盘损坏或操作员操作失误等原因会导致数据丢失、不完整或数据错误，此时，需要对数据库进行还原，将数据还原到某一天，前提是当天必须进行了数据备份。

10.4.2　数据库备份

（1）打开企业管理器，展开“数据库”节点，选中欲备份的数据库，如图 10.33 所示。

（2）在图 10.33 中单击“添加”按钮，打开“选择备份目的”对话框，为数据库指定备份文件，具体操作如图 10.34 所示。

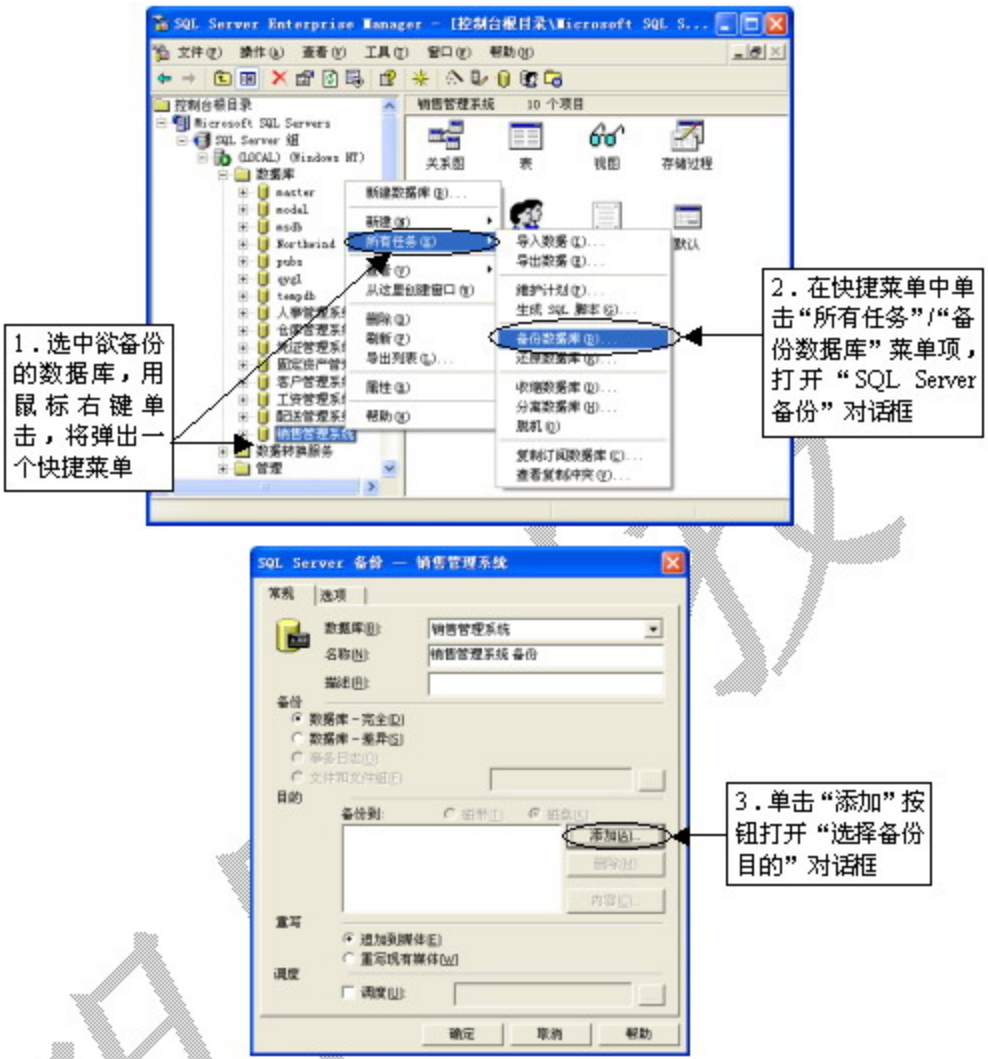


图 10.33　打开数据库备份对话框

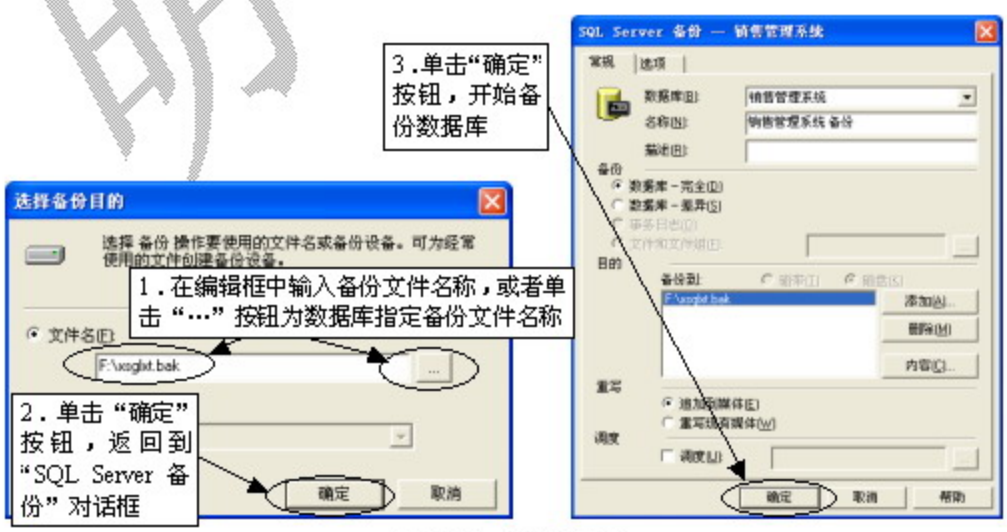


图 10.34　备份数据库

10.4.3 数据库还原

打开企业管理器，展开“数据库”节点，选中欲还原的数据库，具体操作如图 10.35 所示。

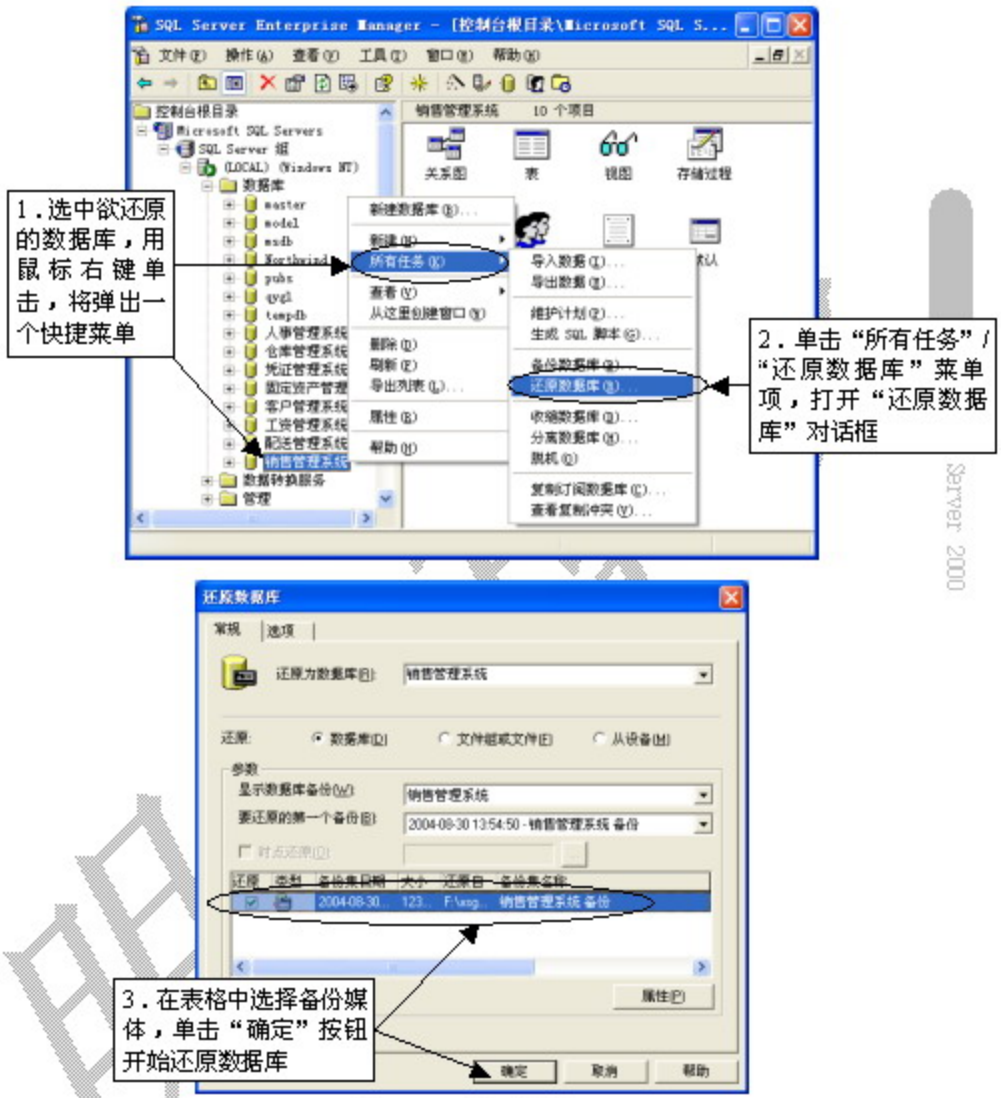


图 10.35 还原数据库

10.5 分离和附加数据库

10.5.1 分离数据库

分离数据库是将数据库从服务器中分离出去，但并没有删除数据库，数据库文件依然存在，如果在需要使用数据库时，可以通过附加的方式将数据库附加到服务器中。在 SQL Server 2000 中分离数据库非常简单，具体方法如下。

打开企业管理器，展开“数据库”节点，选中欲分离的数据库，具体操作如图 10.36 所示。

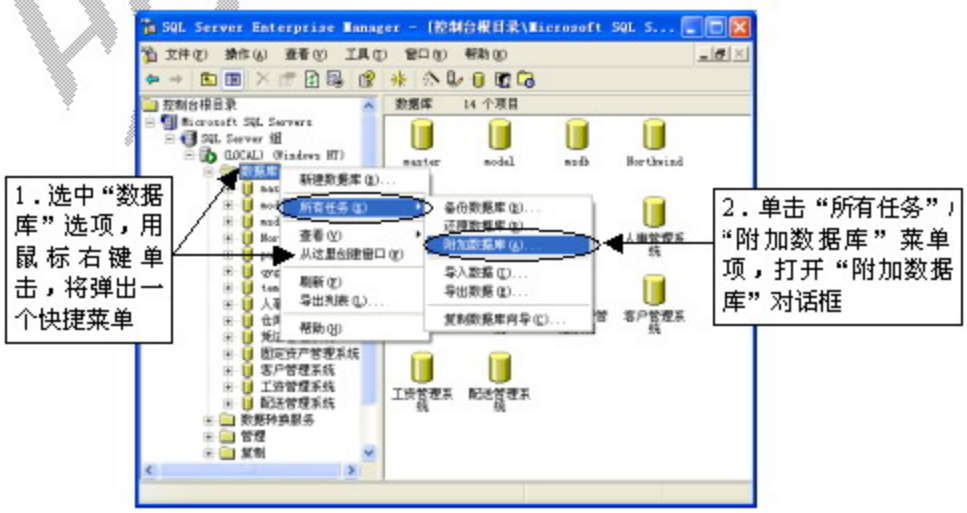


图 10.36　分离数据库

10.5.2　附加数据库

通过附加方式可以向服务器中添加数据库，前提是需要存在数据库文件和数据库日志文件。下面以附加 10.5.1 小节中分离的数据库为例介绍如何附加数据库。

打开企业管理器，用鼠标右键单击“数据库”选项，将弹出一个快捷菜单，具体操作如图 10.37 所示。



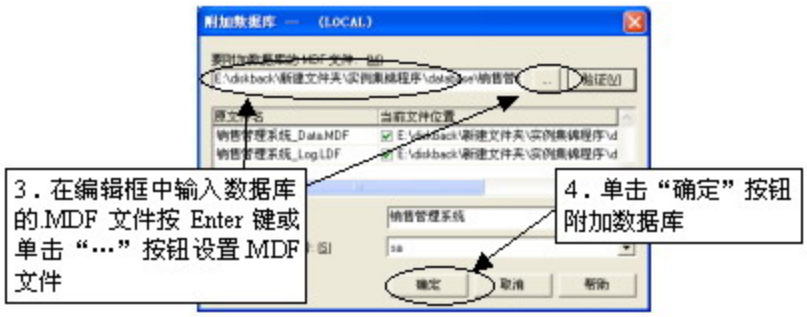


图 10.37　附加数据库

10.6　导入、导出数据表

10.6.1　导入数据库

在 SQL Server 2000 中，用户可以将其他服务器中的数据库或数据表导入到自己的系统中，而且在导入过程中可以选择自己需要的数据表，将其导入到系统中，而不必将所有的数据表都导入到系统中。

10.6.2　导入 SQL Server 数据表

在 SQL Server 2000 中导入数据非常方便，下面以向“销售管理系统”数据库中导入“人事管理系统”数据库中的“员工基础信息表”为例介绍如何导入数据。

（1）打开企业管理器，用鼠标右键单击“数据库”选项，弹出如图 10.38 所示的菜单。



图 10.38　打开导入/导出向导

（2）选择数据源（本例为“人事管理系统”）和目标数据库（本例为“销售管理系统”），如图 10.39 所示。

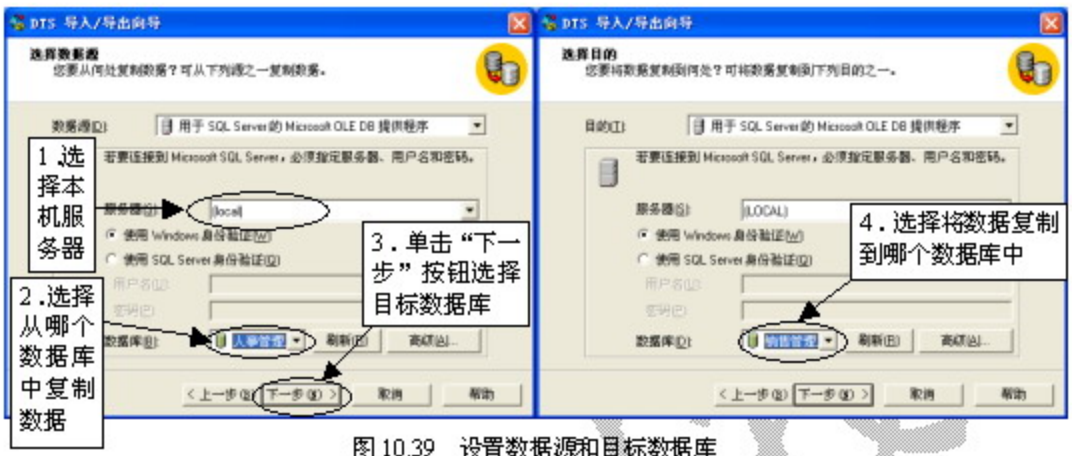


图 10.39　设置数据源和目标数据库

（3）选择数据表，复制数据。在图 10.39 右图中单击“下一步”按钮，选择复制的数据库对象，具体操作如图 10.40 所示。

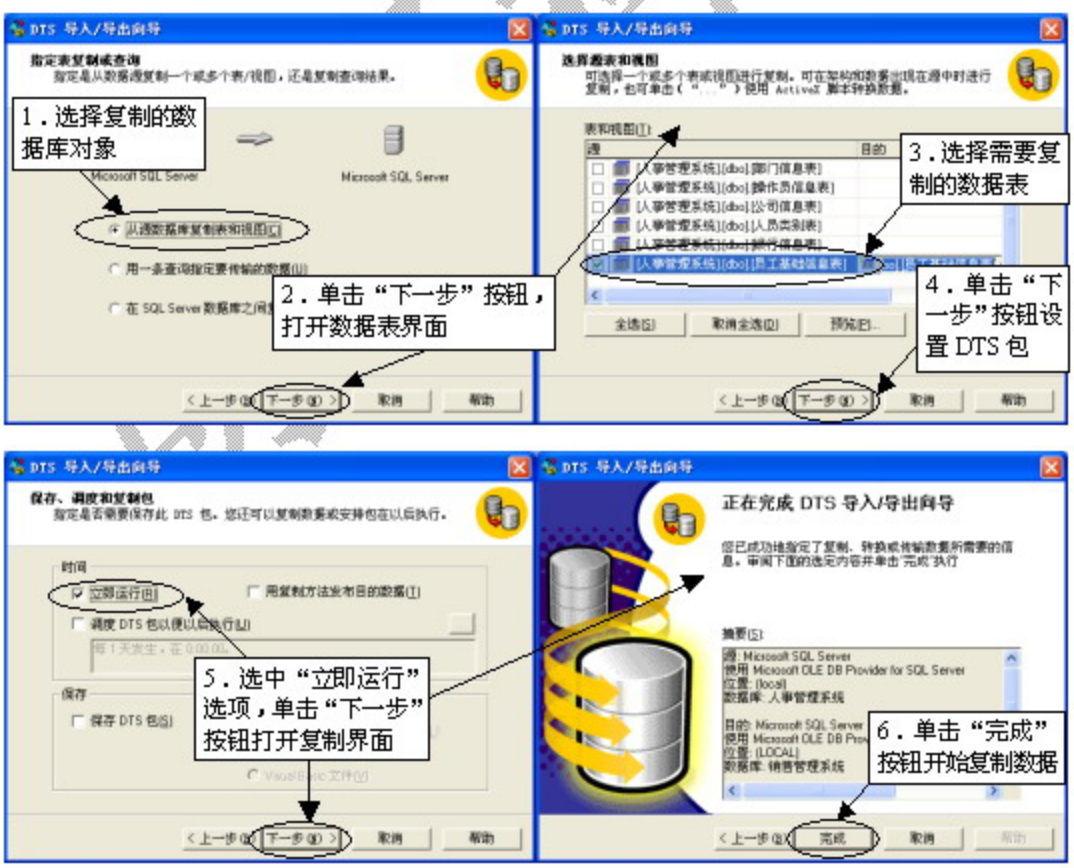


图 10.40　开始复制数据

10.6.3 导入其他数据源的数据表

使用 SQL Server 导入/导出工具还能够将其他数据库中的数据表导入到 SQL Server 中。

（1）打开企业管理器，用鼠标右键单击“数据库”选项，弹出如图 10.41 所示的菜单。



图 10.41　打开导入/导出向导

（2）单击“下一步”按钮，设置数据源，这里选择一个 Access 数据库作为数据源，具体操作如图 10.42 所示。



图 10.42　选择数据源

（3）单击“下一步”按钮，设置目标数据库，这里选择 SQL Server 中的“人事管理系统”，具体操作如图 10.43 所示。

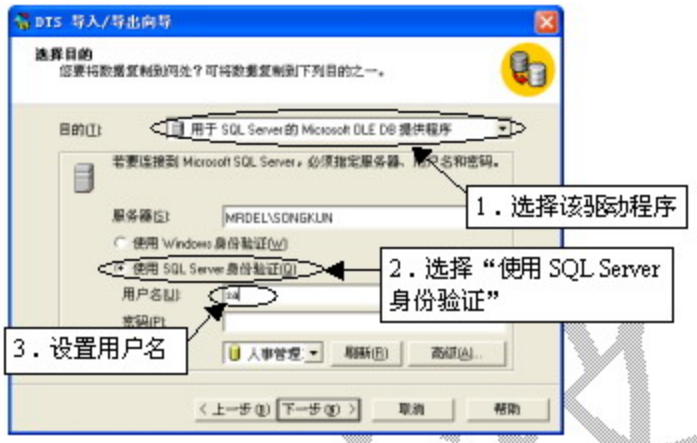


图 10.43　设置目标数据源

（4）单击“下一步”按钮进入“指定表复制或查询”界面，如图 10.44 所示。



图 10.44　指定表复制或查询

（5）保持默认设置，单击“下一步”按钮进入“选择源表和视图”界面，如图 10.45 所示。

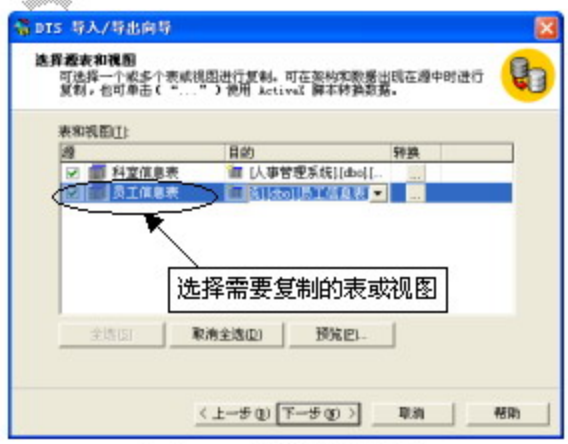


图 10.45　选择源表和视图

（6）单击“下一步”按钮，进入“保存、调度和复制包”界面，如图 10.46 所示。



图 10.46　保存、调度和复制包

（7）单击“下一步”按钮进入“正在完成 DTS 导入/导出向导”界面，如图 10.47 所示。



图 10.47　正在完成 DTS 导入/导出向导

（8）单击“完成”按钮开始导入数据。在数据导入完成后，“人事管理系统”数据库中将出现新的数据表，如图 10.48 所示。

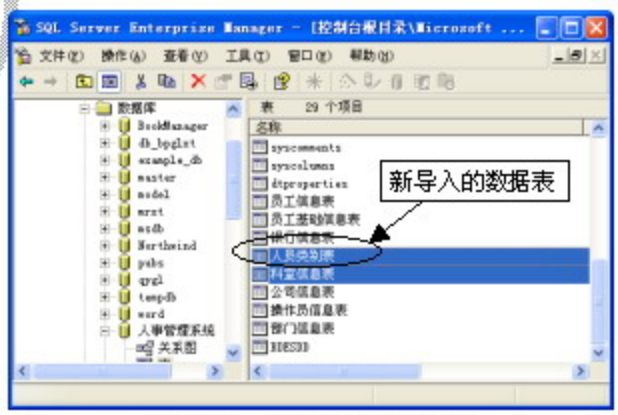


图 10.48　人事管理系统数据表

10.6.4 导出数据库

在 SQL Server 2000 中可以实现将本地服务器或远程服务器中的数据导出到另一个服务器中，它与导入数据是相对的。

10.6.5 导出 SQL Server 数据表

下面的例子将“人事管理系统”数据库中的“人员类别表”导出到“销售管理系统”中，具体步骤如下。

打开企业管理器，用鼠标右键单击“数据库”选项，在弹出的快捷菜单中单击“所有任务”选项，在弹出的子菜单中单击“导出数据”菜单项，打开“DTS 导入/导出向导”对话框，进行如图 10.49 所示对话框中的操作，完成导出 SQL Server 数据表的功能。

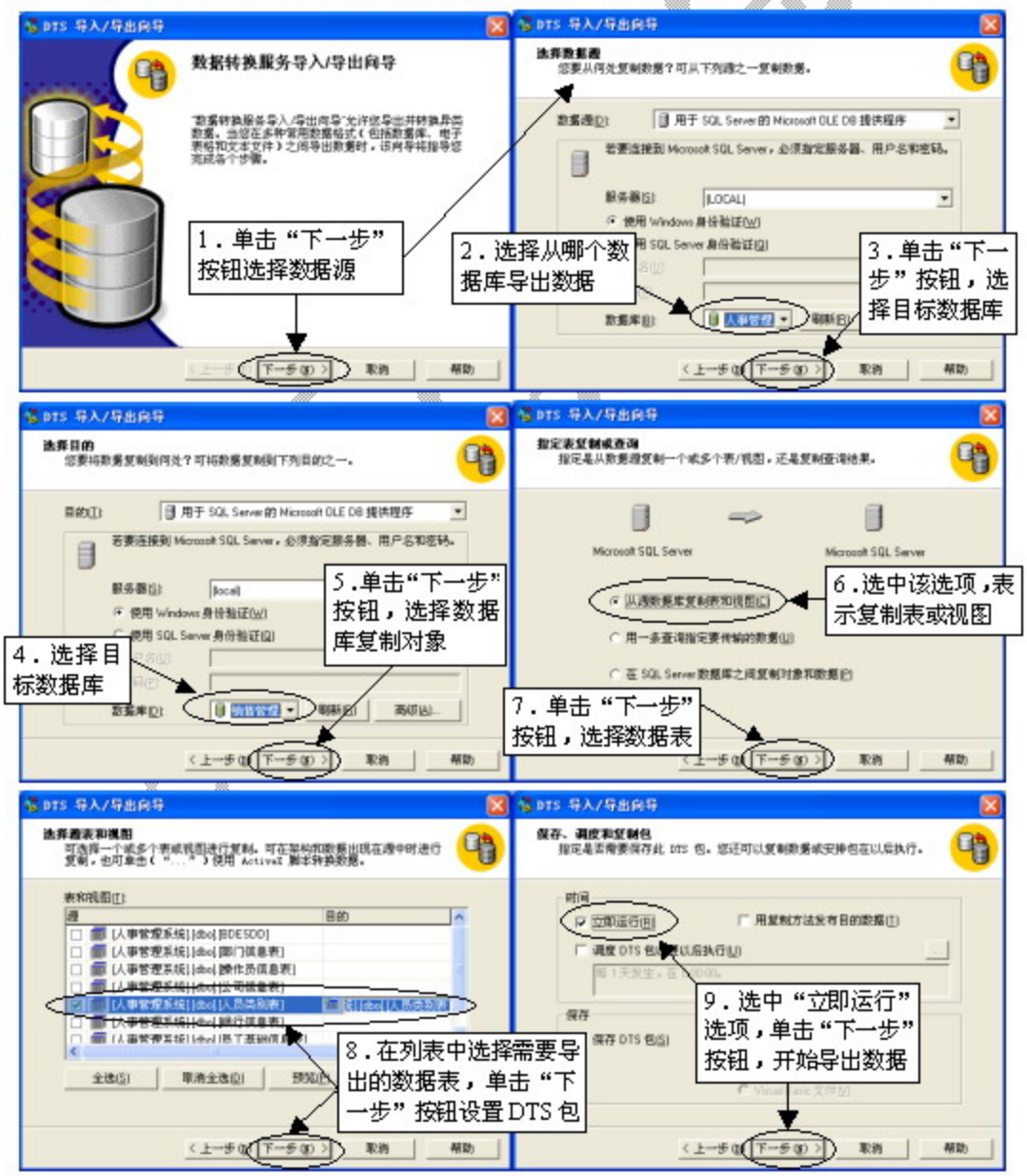




图 10.49　导出数据

明日科技

设计 SQL Server 数据库 第 11 章

11.1 分析数据库

11.1.1 分析数据库结构

数据库设计在开发应用程序中至关重要，合理的数据库结构不仅有利于软件的快速开发，而且有利于以后对软件的维护。下面以商品销售管理系统为例介绍如何分析和设计数据库。在销售管理中需要包含商品信息、库存信息、销售信息、客户信息等，因此，销售管理系统至少需要 4 个数据表，即“商品信息表”、“库存信息表”、“销售表”、“客户信息表”，但在实际应用中，只有这 4 个表是远远不够的。例如，在商品销售时需要记录销售信息，销售信息即包括每种商品的销售信息（商品编号、销售数量、销售金额等），又要包括销售汇总信息（此次销售的总数量、总金额等），只使用一个“销售表”来存储这些信息几乎是不可能实现的，所以还需要使用一个“销售票号表”来记录销售的汇总信息。

在“商品信息表”中记录的是与商品相关的信息，在“库存表”中记录的是商品仓库信息，所以在“库存表”中的商品一定在“商品信息表”中存在，而在“商品信息表”中的商品未必出现在库存表中，“商品信息表”与“库存表”的关系是一对多的关系。

在“销售票号表”中记录的是一次销售操作的汇总信息，而“销售表”中记录的是一次销售操作中每种商品的销售信息，因为在商品销售时，一次可以销售多种商品，所以“销售票号表”与“销售表”的关系是一对多的关系。

在商品销售时，销售的商品应该在库存中存在，而库存中的各种商品未必一次都销售出去，所以，“库存表”与“销售表”为一对多的关系。

在“销售表”中除了记录销售的商品信息，同时还记录了商品销售去向，即客户信息。由于在销售时，一次只能向一个客户销售商品，

所以“客户信息表”与“销售表”的关系为一对多的关系。

根据上述分析，可以得出如图 11.1 所示的模型。

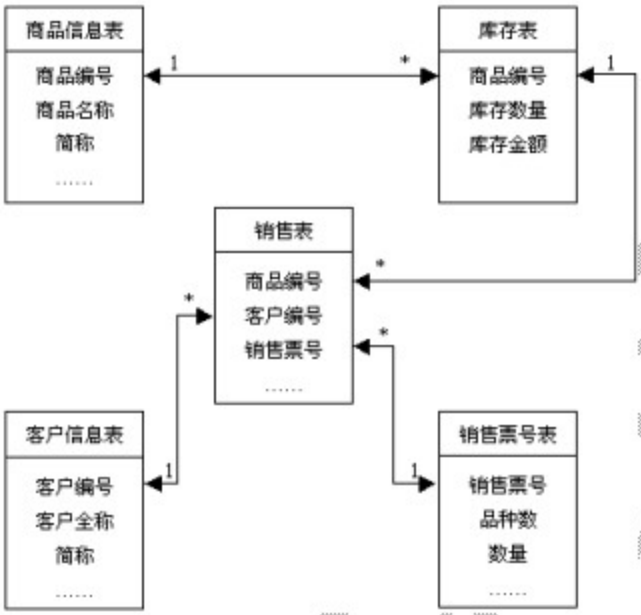


图 11.1　数据模型

11.1.2　设计数据库草稿

在确定应用程序需要使用的数据表及表之间的关系后，接下来的工作便是设计数据表。设计数据表一定要按照预先确定好的数据模型来设计表之间的关系，表 11.1～表 11.5 是作者根据图 11.1 的描述设计的数据表。

表 11.1　　商品信息表

字 段 名 称	字 段 类 型	主　　键	外　　键	惟一索引	是否为空
商品编号	Varchar(30)	是		是	否
商品名称	Varchar(30)				
简称	Varchar(30)				
规格	Varchar(30)				
产地	Varchar(30)				
保质期	Int				
进价	Money				
批发价	Money				
零售价	Money				

表 11.2　　库存表

字 段 名 称	字 段 类 型	主　　键	外　　键	惟一索引	是否为空
商品编号	Varchar(30)		是	是	否
库存数量	Int				
库存金额	Money				

表 11.3　　销售票号表

字段名称	字段类型	主　键	外　键	惟一索引	是否为空
销售票号	Varchar(30)	是		是	否
品种数	Int				
数量	Int				
应付金额	Money				
付款方式	Varchar(30)				
实付金额	Money				
销售时间	Datetime				

表 11.4　　客户信息表

字段名称	字段类型	主　键	外　键	惟一索引	是否为空
客户编号	Varchar(30)	是		是	否
客户全称	Varchar(30)				
简称	Varchar(30)				
地址	Varchar(30)				
传真	Varchar(30)				
联系人	Varchar(30)				
联系电话	Varchar(30)				
开户银行	Varchar(30)				
纳税人	Varchar(30)				
邮箱地址	Varchar(30)				

表 11.5　　销售表

字段名称	字段类型	主　键	外　键	惟一索引	是否为空
商品编号	Varchar(30)		是		否
客户编号	Varchar(30)		是		
销售票号	Varchar(30)		是		
数量	Int				
金额	Money				
折扣	Money				

11.2　设计数据库

11.2.1　创建数据库

在安装完 SQL Server 2000 后，“服务管理器”会自动添加到系统的“开始”菜单下，同时，在任务栏中会出现它的图标，双击该图标会打开“服务管理器”，如图 11.2 所示。

只有服务器处于启动状态，用户才能使用 SQL Server 2000 提供的服务，所以，在使用 SQL Server 2000 之前，需要启动服务器。通常情况下，安装完 SQL Server 2000 后，服务器会自动启动，并且，如果选中“当启动 OS 时自动启动服务”选项，每当操作系统（Windows）启动时，服务器就会自动启动。

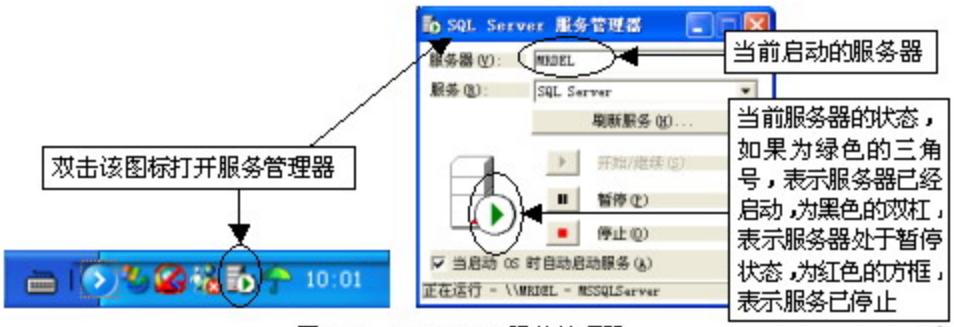


图 11.2　SQL Server 服务管理器

在开发程序时，通常离不开数据库，数据库可以认为是应用程序后端的服务器，提供对数据进行存储的能力。下面以创建“销售管理系统”数据库为例介绍如何在 SQL Server 2000 中创建数据库。

（1）打开 SQL Server 企业管理器，展开各个节点，如图 11.3 所示。

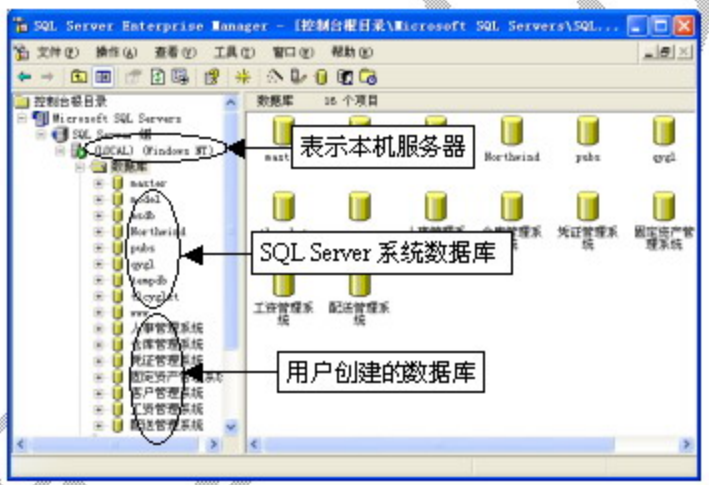


图 11.3　企业管理器

（2）选中“数据库”选项，用鼠标右键单击会弹出一个快捷菜单，具体操作如图 11.4 所示。

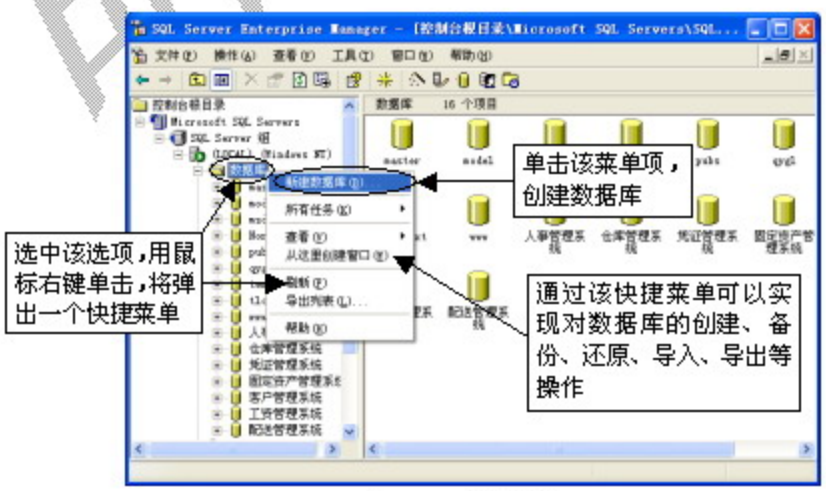


图 11.4　快捷菜单

（3）单击“新建数据库”菜单项，弹出如图 11.5 所示的对话框。

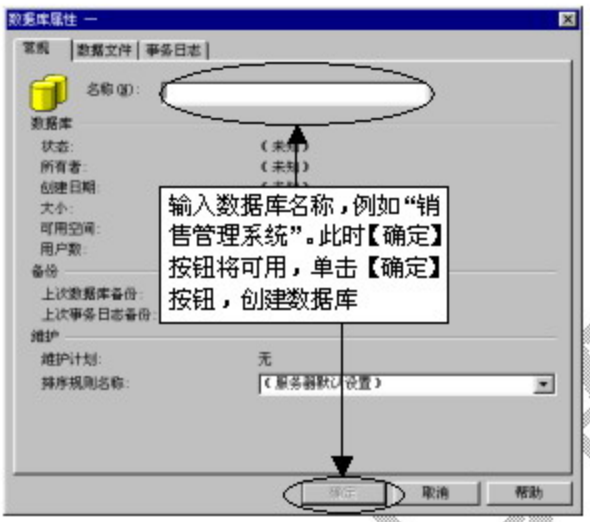


图 11.5　“数据库属性”对话框

（4）在“名称”编辑框中输入所要创建的数据库名称，例如“销售管理系统”。单击“确定”按钮创建数据库，如图 11.6 所示。

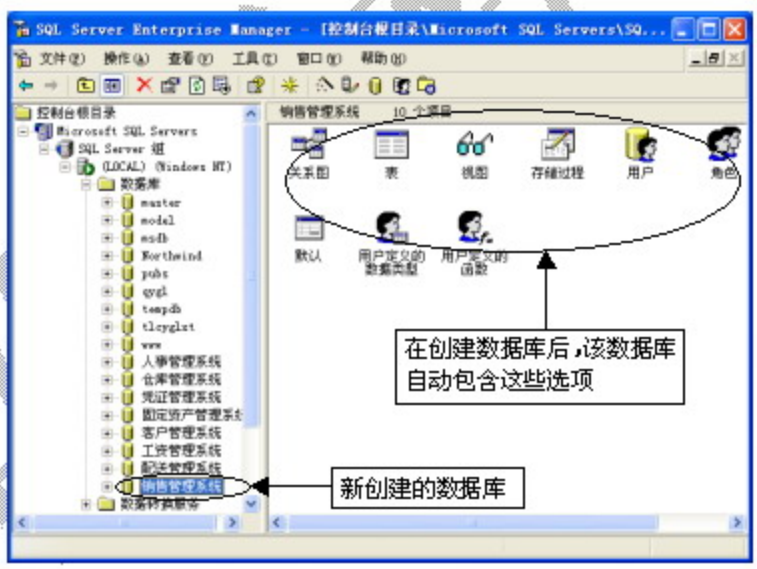


图 11.6　创建数据库

11.2.2　添加数据表

（1）打开企业管理器，在企业管理器中选中一个数据库，例如前面创建的“销售管理系统”数据库，展开该数据库，用鼠标右键单击“表”选项会弹出一个快捷菜单，如图 11.7 所示。

（2）单击“新建表”菜单项，打开如图 11.8 所示的对话框。

（3）在“列名”字段中输入字段名称，在“数据类型”字段中选择字段的数据类型，在“长度”字段中输入字段长度，具体操作如图 11.9 所示。

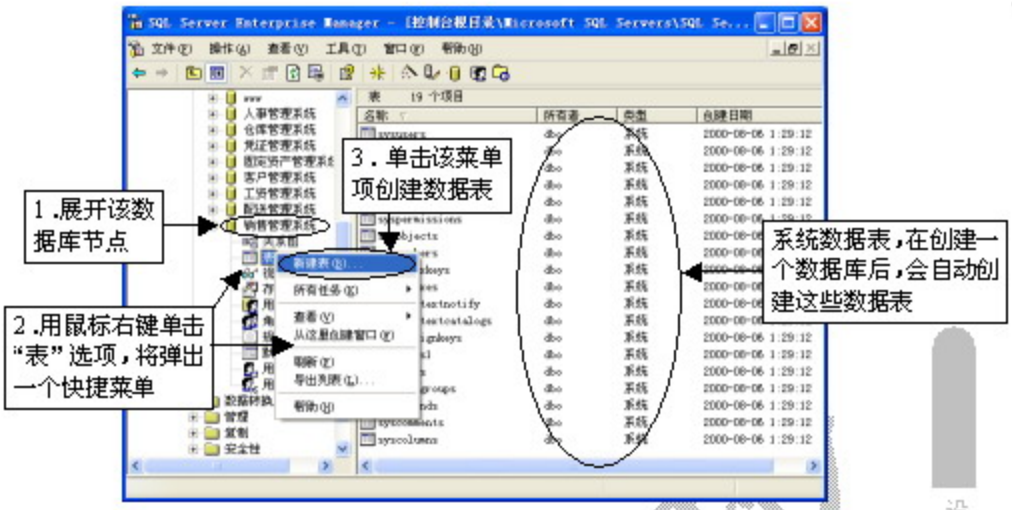


图 11.7　弹出式菜单

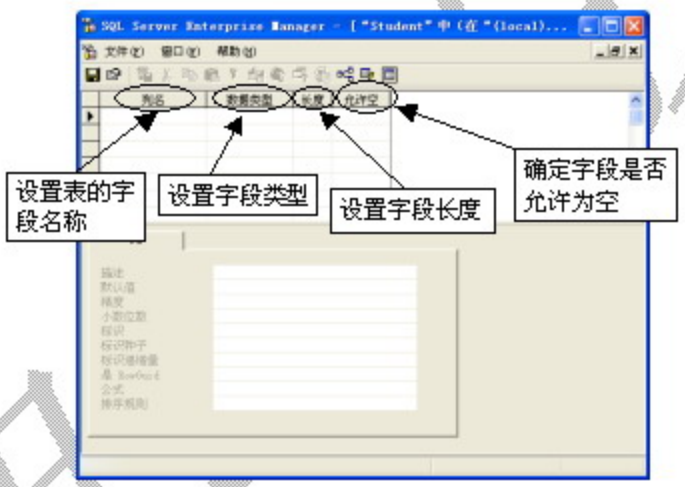


图 11.8　创建数据表

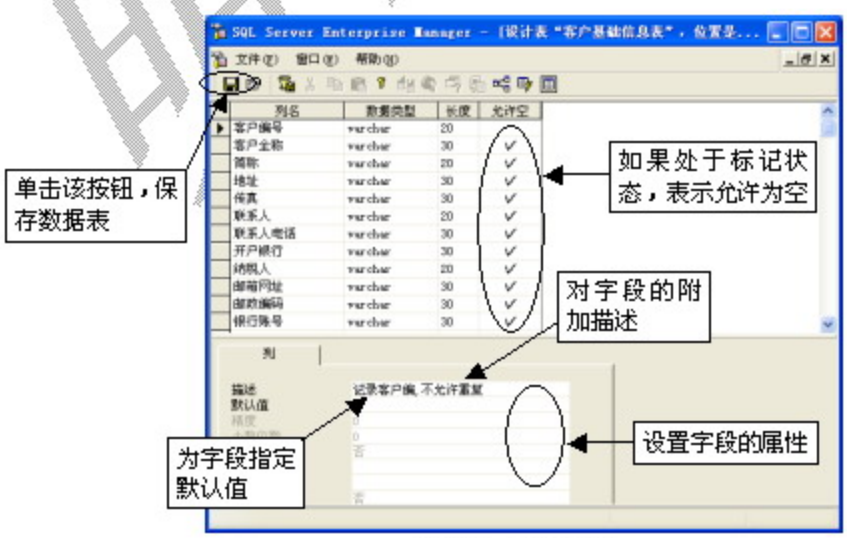


图 11.9　设置表结构

11.2.3 选择数据类型

在创建数据表时，需要为字段设置数据类型，如图 11.10 所示。

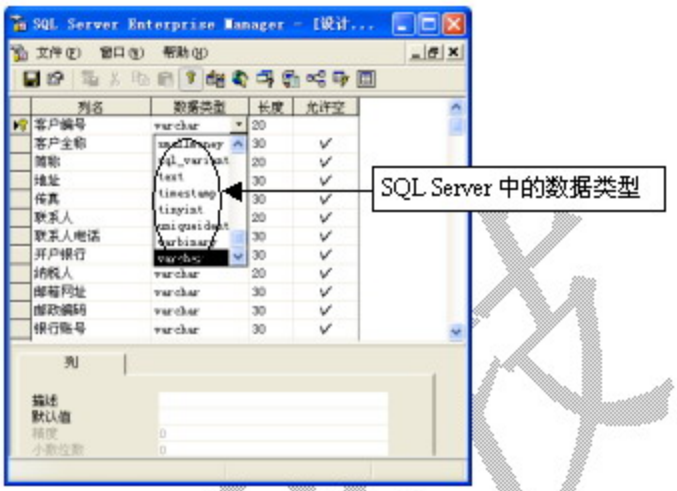


图 11.10 SQL Server 数据类型

在 SQL Server 中，系统共定义了 29 种数据类型，意义如下。

- Bigint

Bigint 为整数类型。当整型值超过 Int 数据类型支持的范围时，将使用 Bigint 数据类型。

- Binary

Binary 为定长的二进制类型，其最大长度为 8000 个字节。

- Bit

Bit 表示 0 或 1 的整数数据。

- Char

Char 表示固定长度的字符数据类型，其最大长度为 8000 个字符。

- Cursor

这是变量或存储过程 OUTPUT 参数的一种数据类型，这些参数包含对游标的引用。

- Datetime

Datetime 为日期时间数据类型，表示范围从 1753 年 1 月 1 日到 9999 年 12 月 31 日，精确到 0.03 秒。

- Decimal

Decimal 表示固定精度与小数位数的实型数据。

- Float

Float 为从−1.79E+308 到 1.79E+308 的浮点精度数字。

- Image

Image 表示可变长度的二进制数据，多用于存储图像。

- Int

Int 表示从-2^{31}到$2^{31}-1$之间的整型数据。

- Money

Money 表示实型数据，精确到货币单位的 1%。

- Nchar

Nchar 表示固定长度的 Unicode 数据，最大长度为 4 000 个字符。

- Ntext

Ntext 表示可变长度的 Unicode 数据。

- Numeric

Numeric 与 Decimal 相同。

- Nvarchar

Nvarchar 为可变长度 Unicode 数据，其最大长度为 4 000 字符。

- Real

Real 表示从−3.40E+38 到 3.40E+38 的浮点精度数字。

- Timestamp

Timestamp 表示数据库范围的惟一数字，每次更新行时也要对其进行更新。

- Rowversion

Rowversion 与 Timestamp 同义。

- Smalldatetime

Smalldatetime 表示从 1900 年 1 月 1 日到 2079 年 6 月 6 日的日期和时间数据，精确到分钟。

- Smallint

Smallint 表示从 -2^{15}(−32 768)到 $2^{15}-1$(32 767)的整数数据。

- SmallMoney

SmallMoney 表示货币数据值介于−214 748.3648 与+214 748.3647 之间，精确到货币单位的 1%。

- Sql_variant

Sql_variant 表示一种存储 SQL Server 支持的各种数据类型（Text、NText、Timestamp 和 sql_variant 除外）值的数据类型。

- Sysname

Sysname 表示系统提供的用户定义数据类型，与 Nvarchar(128)同义。

- Table

Table 为一种特殊的数据类型，存储供以后处理的结果集。

- Text

Text 为可变长度的非 Unicode 数据。

- Tinyint

Tinyint 表示从 0 到 255 的整数数据。

- Uniqueidentifier

Uniqueidentifier 为全局惟一标识符(GUID)。

- Varbinary

Varbinary 为可变长度的二进制数据，其最大长度为 8 000 个字节。

- Varchar

Varchar 为可变长度的非 Unicode 数据，最长为 8 000 个字符。

11.2.4 删除数据表

打开企业管理器，展开各个节点，选择表所在的数据库，例如“销售管理系统”。展开

明日科技

“数据库”节点，具体操作如图 11.11 所示。

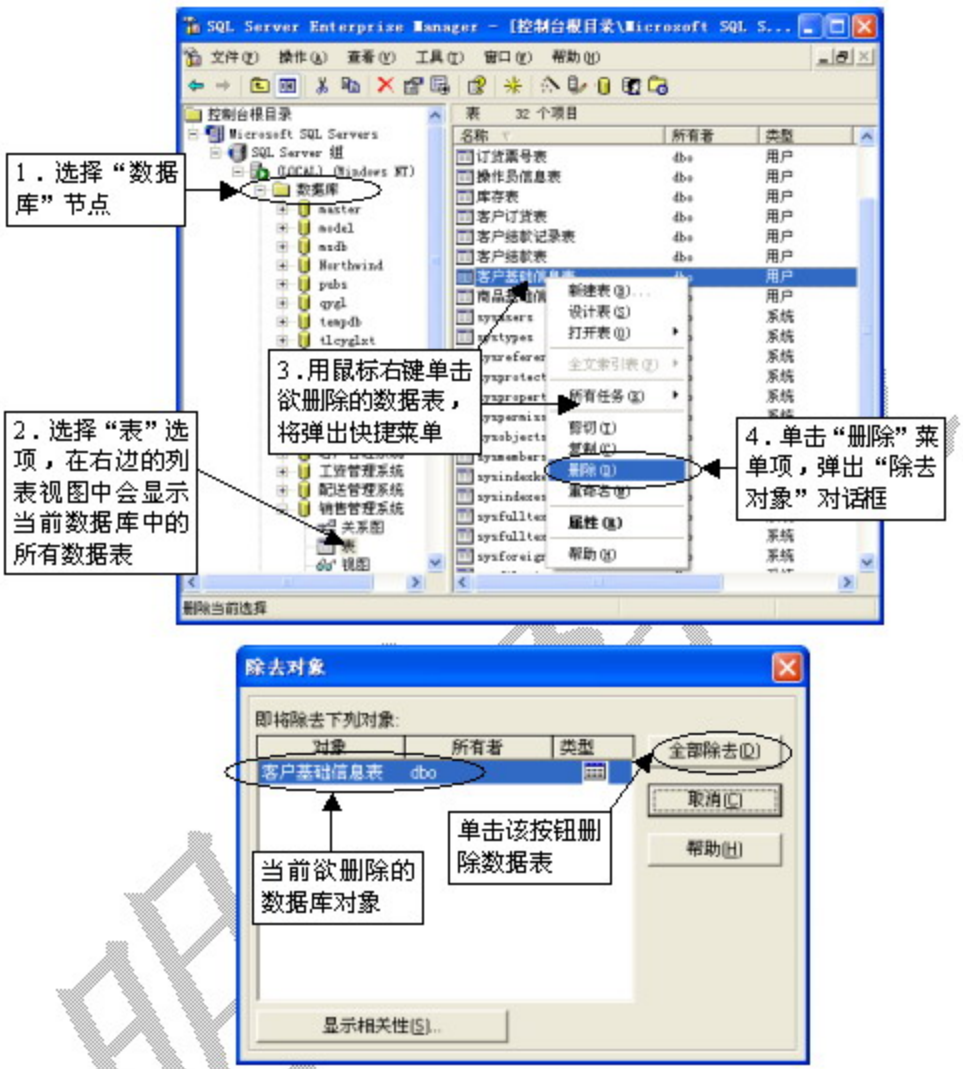


图 11.11　删除数据表

11.3　建立索引

11.3.1　建立惟一索引

当表中某些字段不允许出现重复数据时，可以为这些字段建立惟一索引。向建立惟一索引的字段插入重复数据时会出现错误提示，并且禁止数据插入。下面以具体示例说明如何建立惟一索引。

打开企业管理器，在数据库中选中一个数据表，例如前面创建的“客户基础信息表”，具体操作如图 11.12 所示。

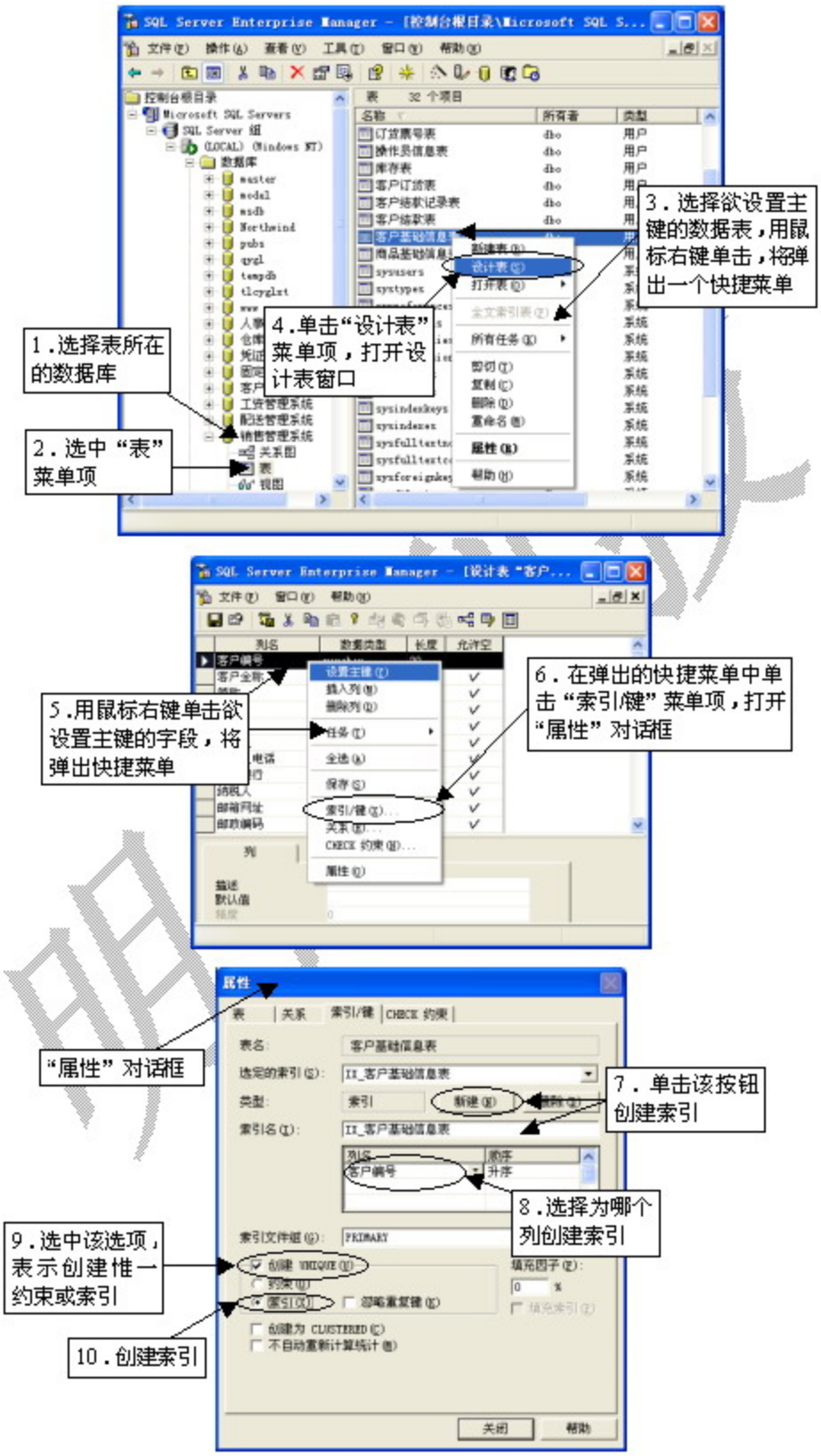


图 11.12　添加惟一索引

11.3.2 定义主键索引

主键惟一标识数据表中所有行的一列或一组列，主键不允许为空，在同一个表中只能有一个主键。定义主键的字段，SQL Server使用创建惟一索引的方式强制数据的惟一性。下面以为“客户基础信息表”的“客户编号”字段创建主键索引为例介绍创建主键索引的方法。

打开企业管理器，在列表视图中选中“客户基础信息表”，具体操作如图 11.13 所示。

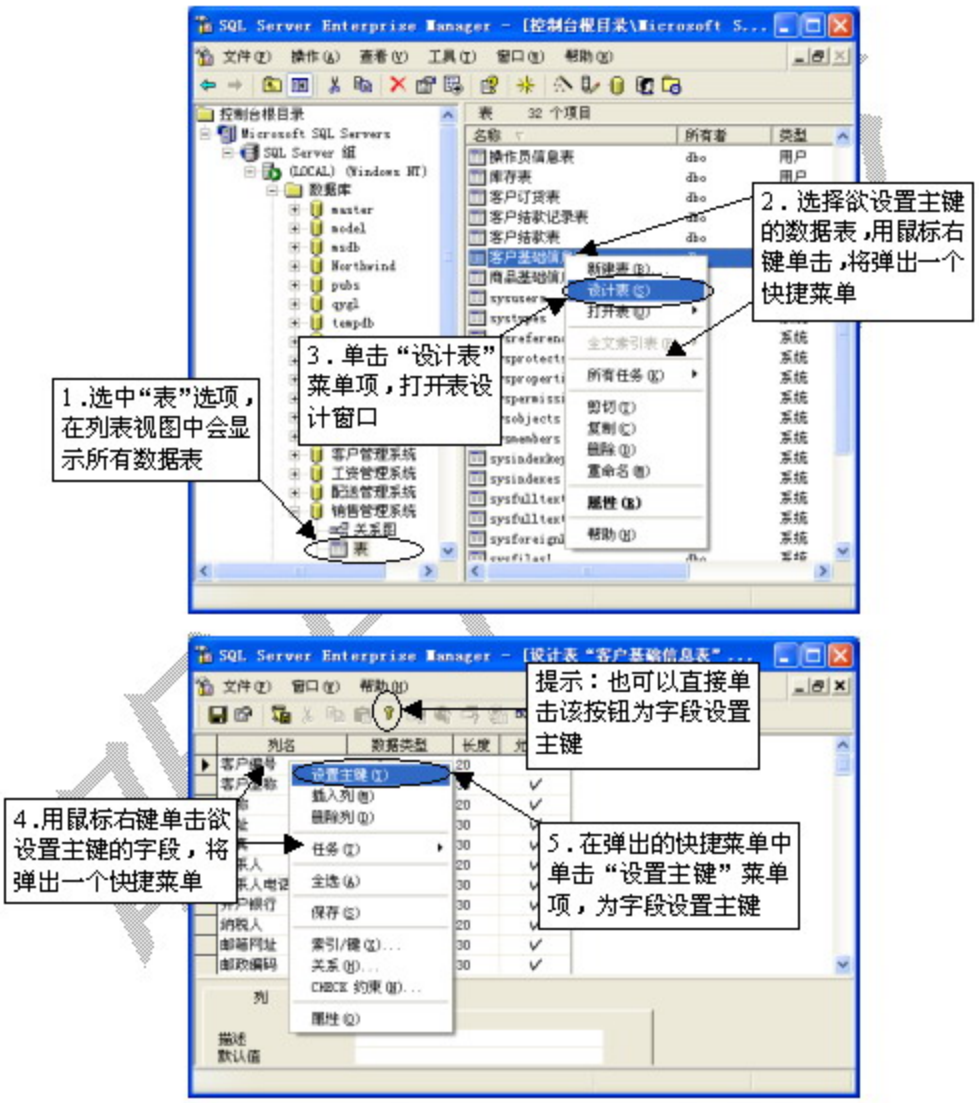


图 11.13 定义主键

11.3.3 删除索引

在 SQL Server 中删除索引非常方便，下面以删除“客户基础信息表”中的索引为例介绍如何删除索引。

在企业管理器中打开“客户基础信息表”表设计窗口，具体操作如图 11.14 所示。

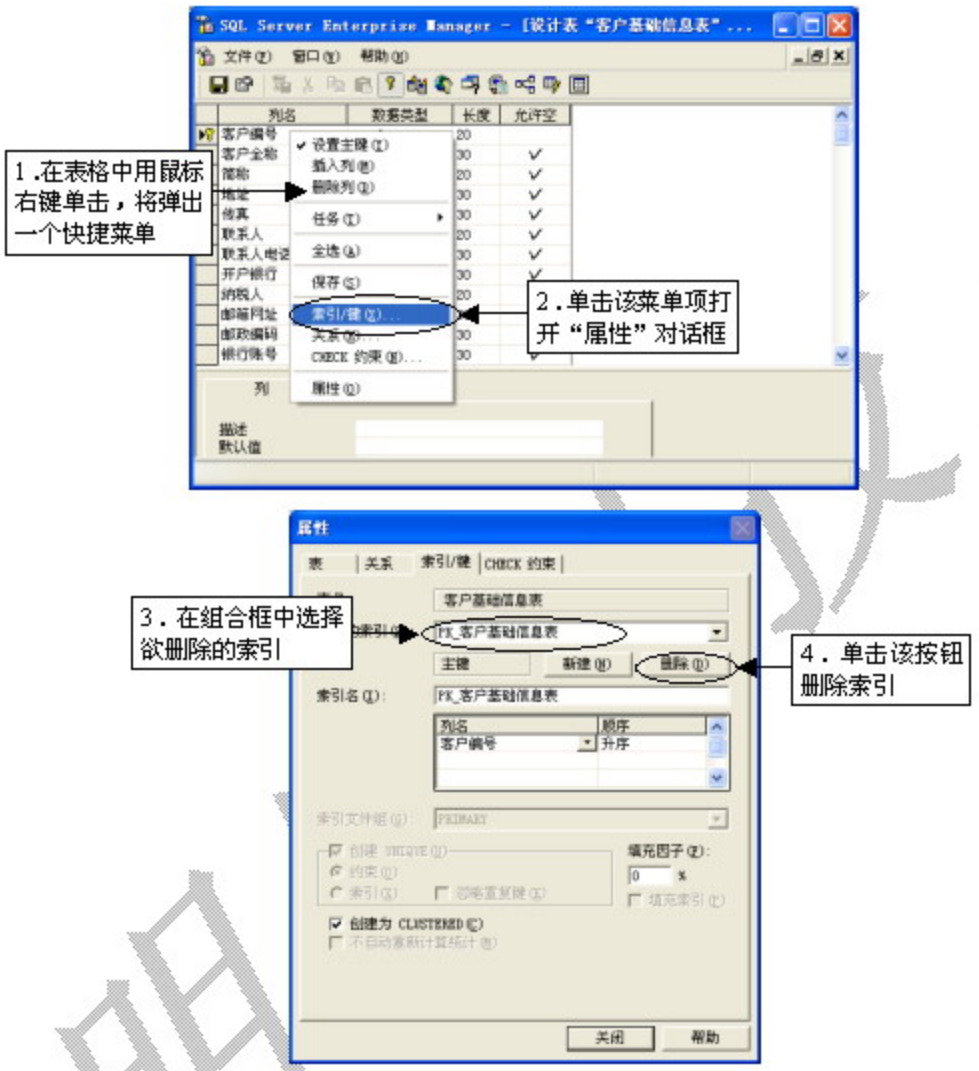


图 11.14　删除索引

注意：在删除一个主键索引时，如果其他表中的字段与主键索引字段有关联，则不允许删除主键索引，除非先解除与其他表的关联。

11.4　修改数据库

11.4.1　为数据表更名

打开企业管理器，展开数据库选项，选中“表”选项，在列表视图中选中欲更名的数据表，具体操作如图 11.15 所示。

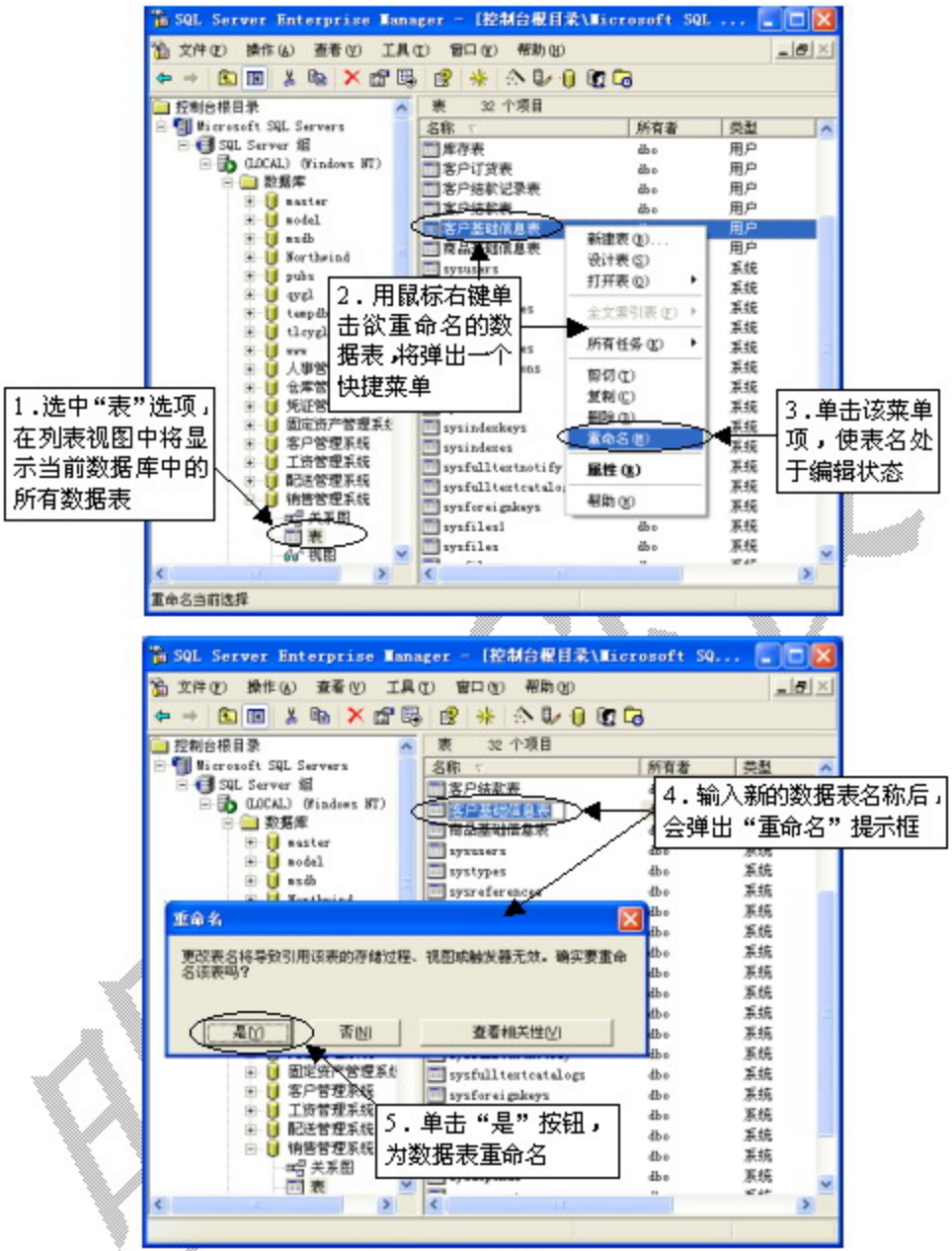


图 11.15　重命名数据表

11.4.2　在表结构中添加新字段

打开企业管理器，展开“数据库”选项，在视图列表中选中欲修改的数据表，具体操作如图 11.16 所示。

11.4.3　在表结构中删除字段

打开企业管理器，展开“数据库”选项，选择欲删除字段的数据表，打开其表设计窗口，具体操作如图 11.17 所示。

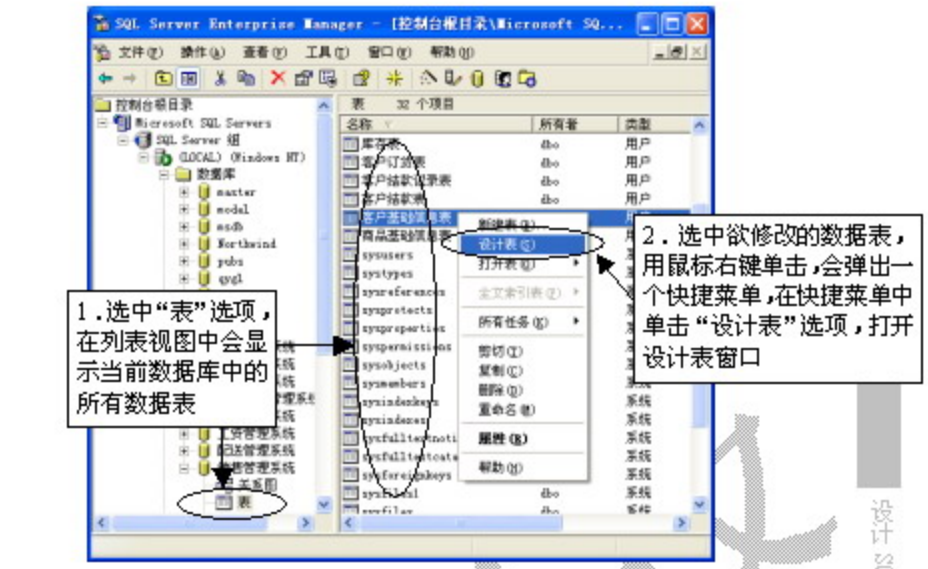


图 11.16　添加字段

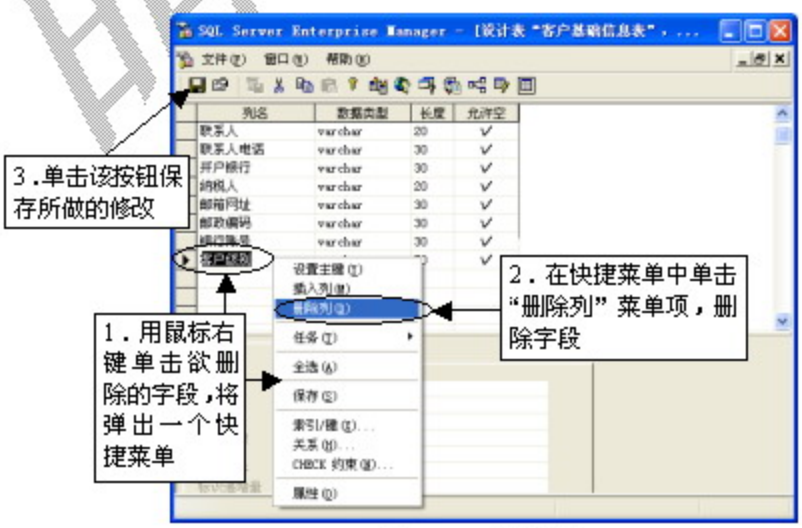


图 11.17　删除字段

11.4.4 改变表结构字段的大小

打开企业管理器，展开“数据库”选项，选中欲修改的数据表，打开其表设计窗口，具体操作如图 11.18 所示。

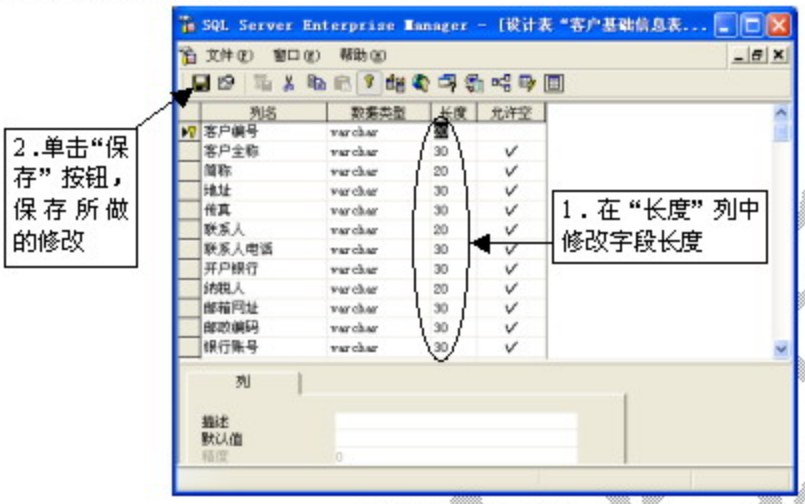


图 11.18 修改字段大小

11.4.5 修改表结构中的字段顺序

打开企业管理器，展开“数据库”选项，在视图列表中选中欲修改的数据表，具体操作如图 11.19 所示。

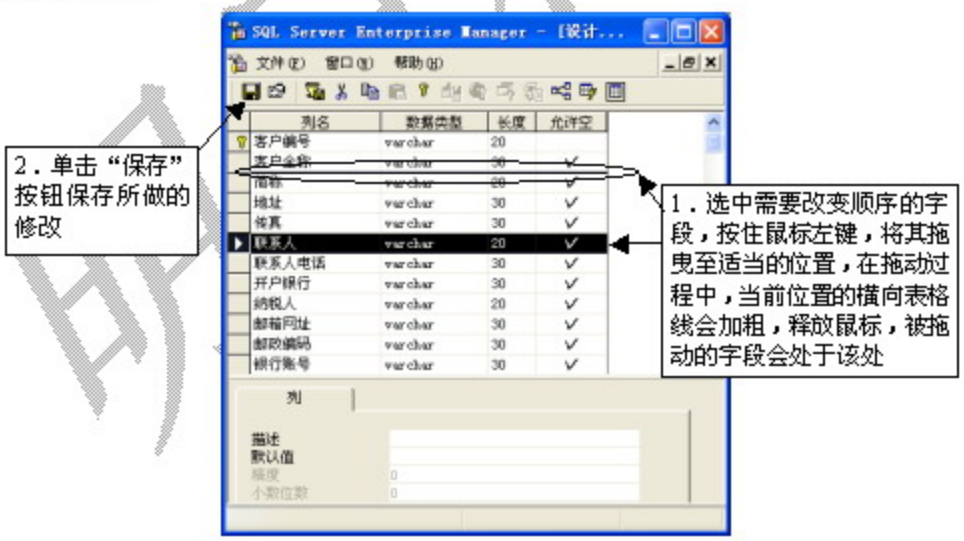


图 11.19 改变字段顺序

11.5 关系的建立与维护

11.5.1 建立一对一关系

一对一关系是指对于集合 A 中的每一个元素，在集合 B 中至多只有一个元素与之对应，

反之，对于集合 B 中的每一个元素，在集合 A 中至多也只有一个元素与之对应，如图 11.20 所示。

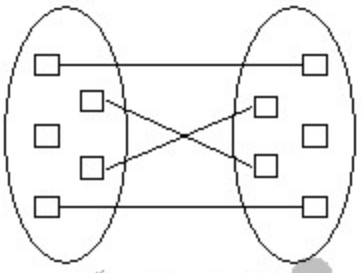


图 11.20　一对一关系

在设计数据库时，表之间的关系表现为主键表与外键表之间的关系，即主表与从表之间的关系，所以在设计表关系时，首先需要确定主表与从表。下面以建立“客户基础信息表”和“客户结款表”的一对一关系说明如何创建一对一关系。

在“客户基础信息表”中需要存放客户的基础信息，所以，将“客户基础信息表”设计为主表，“客户结款表”设计为从表。表结构如表 11.6、表 11.7 所示。

表 11.6　　客户基础信息表

字 段 名 称	字 段 类 型	主　　键	外　　键	惟一索引	是否为空
客户编号	Varchar(20)	是		是	否
客户全称	Varchar(30)				
简称	Varchar(20)				
地址	Varchar(30)				
传真	Varchar(30)				
联系人	Varchar(20)				
联系人电话	Varchar(30)				
开户银行	Varchar(30)				
纳税人	Varchar(20)				
邮箱地址	Varchar(30)				
邮政编码	Varchar(30)				
银行账号	Varchar(30)				

表 11.7　　客户结款表

字 段 名 称	字 段 类 型	主　　键	外　　键	惟一索引	是否为空
客户编号	Varchar(20)		是	是	否
客户全称	Varchar(30)				
金额小计	Money				
欠款记账	Money				
付款合计	Money				
结款时间	Datetime				

在建立关系之前，需要将主表的关联字段设置为主键，为从表的关联字段创建惟一索引，这是因为创建的是一对一关系。

打开企业管理器，展开“数据库”选项，在列表视图中选中“客户基础信息表”，具体操作如图 11.21 所示。

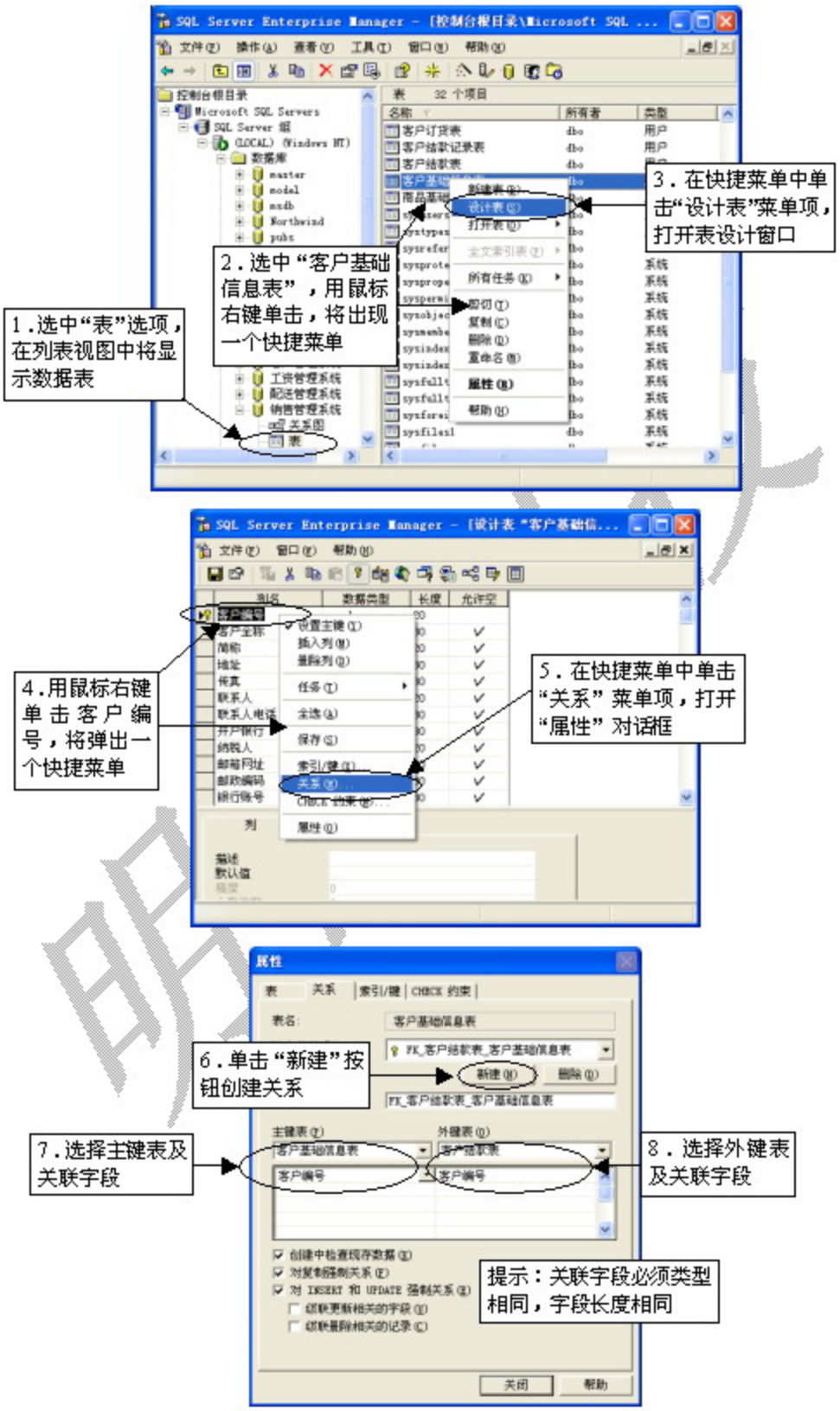


图 11.21　建立一对一关系

11.5.2 建立一对多关系

一对多关系是指对于集合 A 中的每一个元素，在集合 B 中有 N 个元素与之对应（N≥0），反之，对于集合 B 中的每一个元素，在集合 A 中至多有一个元素与之对应，如图 11.22 所示。

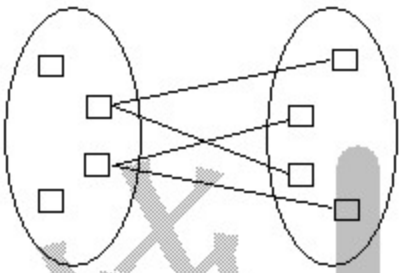


图 11.22 一对多关系

下面以建立“客户结款表”和“客户结款记录表”的一对多关系说明如何创建一对多关系。

在“客户结款表”中对应一个供应商只有一条记录，而在“客户结款记录表”中，由于对一个客户需要进行多次结款，所以，在“客户结款记录表”中对于一个客户，可以有多条记录。综上分析，可以将“客户结款表”设置为主表，将“客户结款记录表”设置为从表，关联字段为“客户编号”。

打开企业管理器，展开“数据库”节点，在“销售管理系统”数据库中选中“表”选项，在列表视图中选中“客户结款表”，用鼠标右键单击，在弹出的快捷菜单中单击“设计表”菜单项，打开表设计窗口，具体操作如图 11.23 所示。

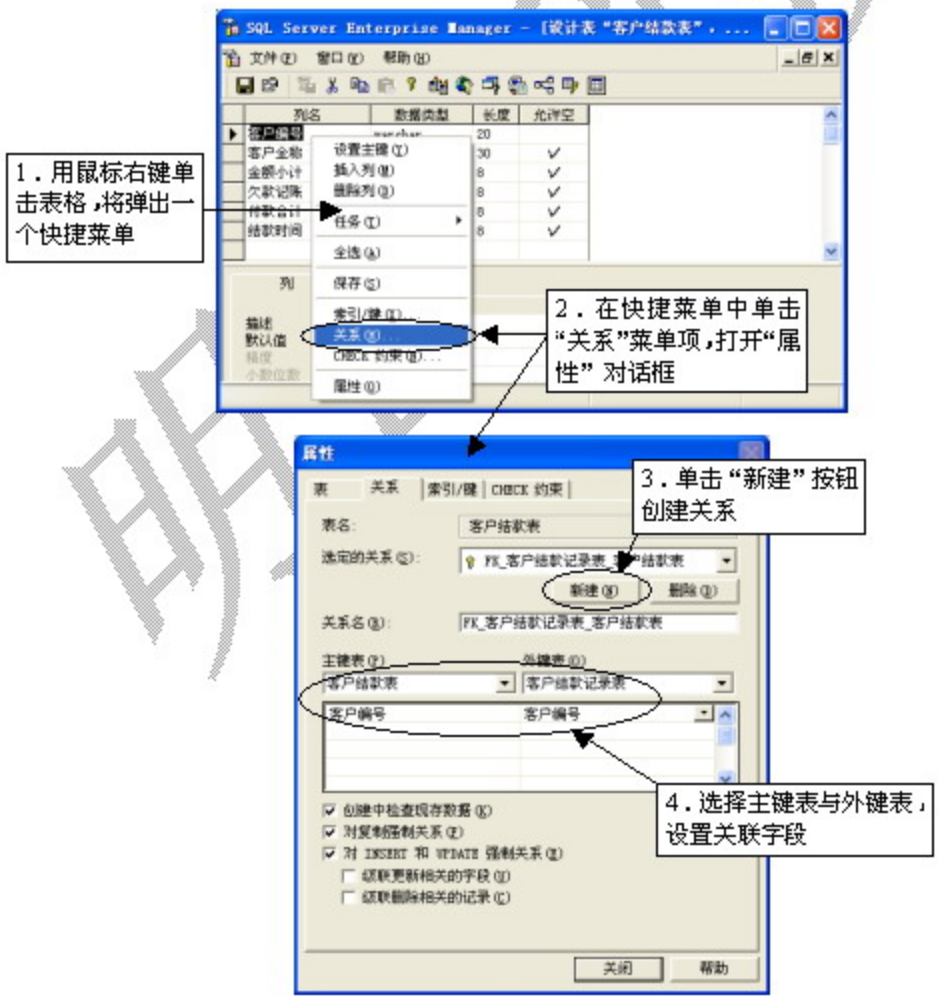


图 11.23 建立一对多关系

11.5.3 关系更名

打开企业管理器，展开“数据库”节点，在“销售管理系统”数据库中选中“表”选项，在列表视图中选中具有关系的数据表（本例为“客户结款表”），可以是主键表，也可以是外键表，用鼠标右键单击，在弹出的快捷菜单中单击“设计表”菜单项打开表设计窗口，具体操作如图 11.24 所示。

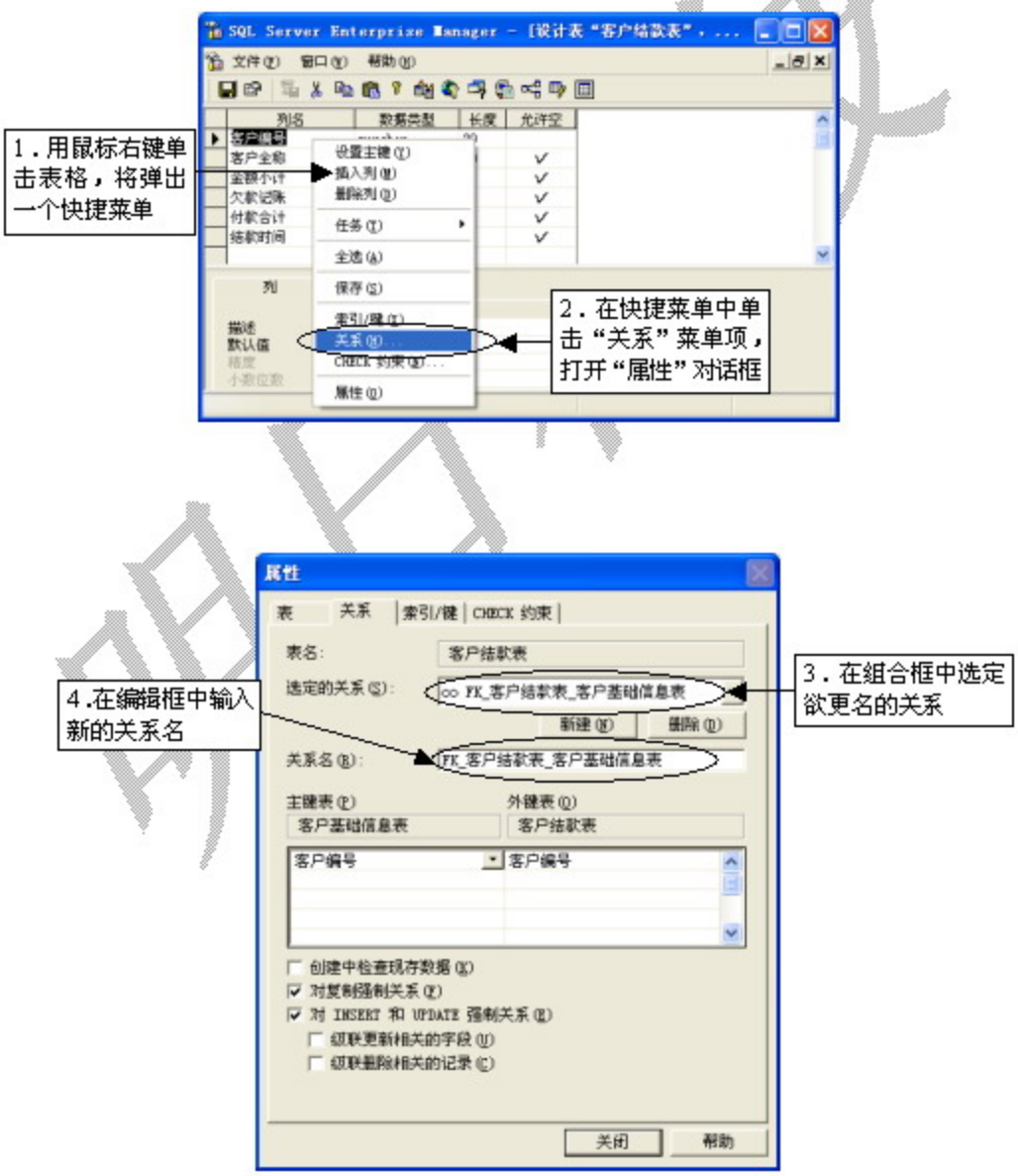


图 11.24　关系更名

11.5.4 删除关系

打开企业管理器，展开“数据库”节点，在“销售管理系统”数据库中选中“表”选项，在列表视图中选中具有关系的数据表（本例为“客户结款表”），可以是主键表，也可以是外键表，用鼠标右键单击，在弹出的快捷菜单中单击“设计表”菜单项打开表设计窗口。在表设计窗口中，用鼠标右键单击表格，在弹出的快捷菜单中单击“关系”菜单项，打开属性窗口，具体操作如图 11.25 所示。

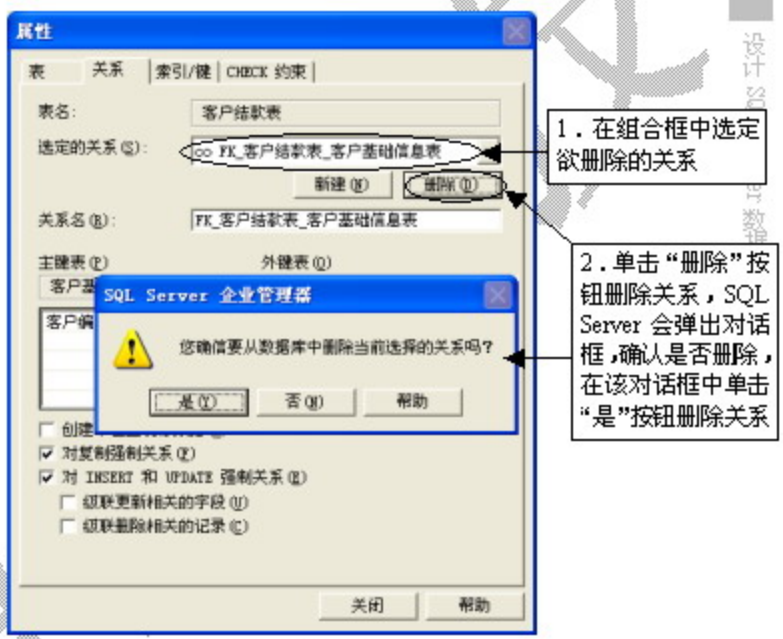


图 11.25 删除关系

11.6 设计视图

11.6.1 创建视图

视图是存在于数据库中的“虚拟表”，其内容是执行查询语句后所得到的查询结果。视图能够从多个表中提取数据，并用单个表显示提取的数据，所以，使用视图可以简化查询。下面以在企业管理器中创建视图为例介绍如何创建视图。

（1）打开企业管理器，展开“数据库”节点，选择一个数据库，例如“人事管理系统”。展开该节点，选中“视图”选项，打开视图设计窗口，具体操作如图 11.26 所示。

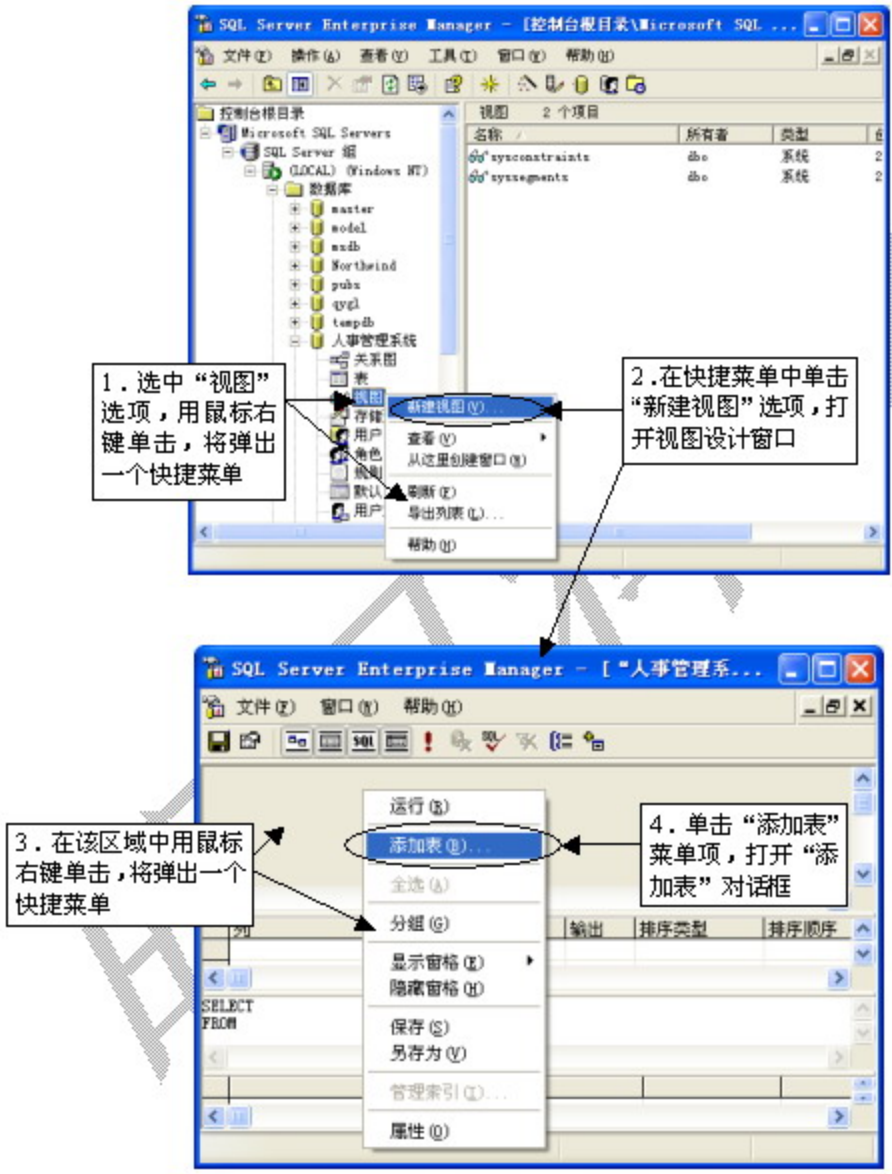


图 11.26　视图设计窗口

（2）通过视图设计器向视图中添加基表，本例向视图中添加两个基表，分别为“部门信息表”和“员工基础信息表”。在图 11.26 中，单击“添加表”菜单项将打开“添加表”对话框，具体操作如图 11.27 所示。

（3）为视图选择字段，在“部门信息表”中选中“部门名称”、“部门电话”、“备注”字段，在“员工基础信息表”中选中“所有列”，具体操作如图 11.28 所示。

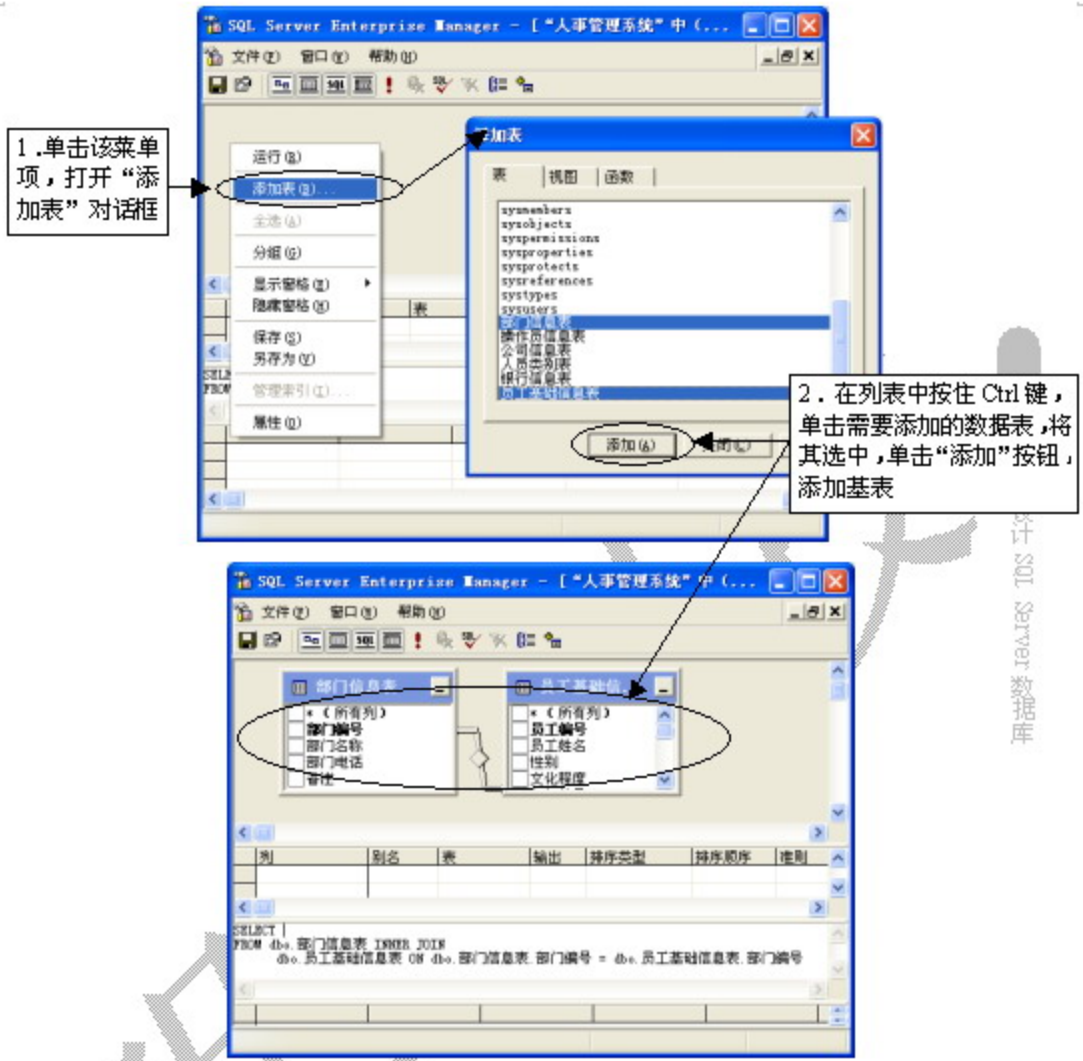


图 11.27　添加基表

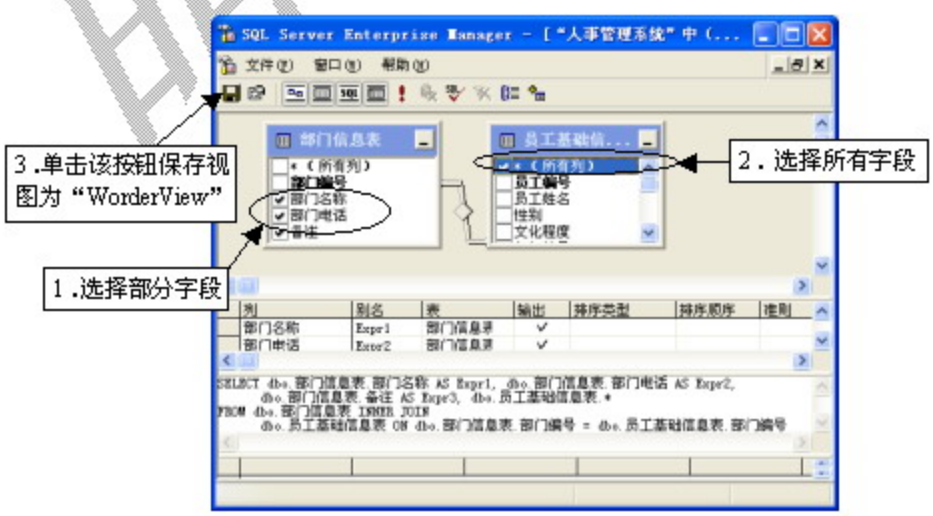


图 11.28　选择视图字段

11.6.2 修改视图

打开企业管理器，展开“数据库”节点，选中一个数据库，例如“人事管理系统”，选中“视图”选项，在右边的列表视图中会列出当前数据库中的所有视图，具体操作如图 11.29 所示。

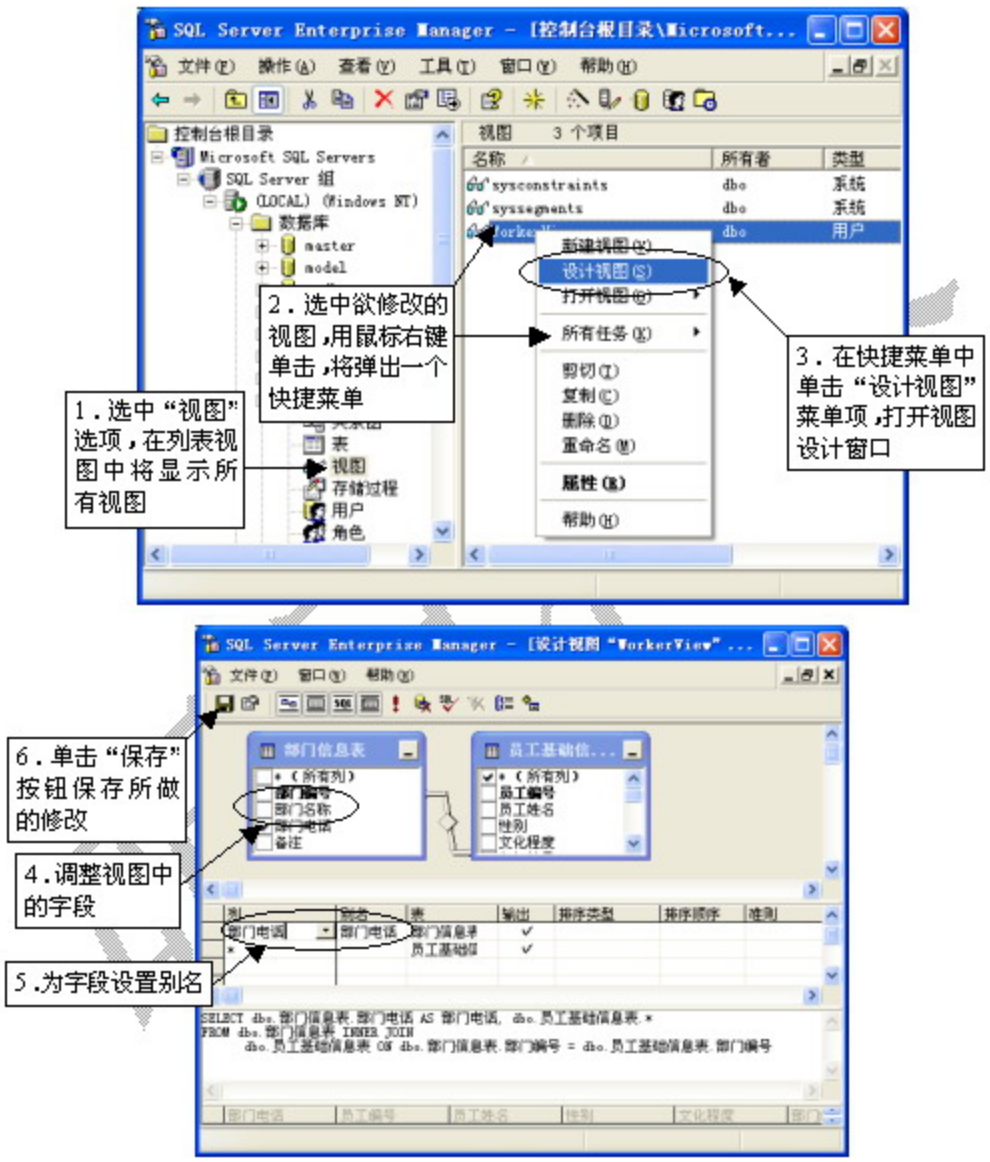


图 11.29 修改视图

11.6.3 删除视图

打开企业管理器，展开“数据库”节点，选中一个数据库，例如“人事管理系统”，选中“视图”选项，在右边的列表视图中会列出当前数据库的所有视图，具体操作如图 11.30 所示。

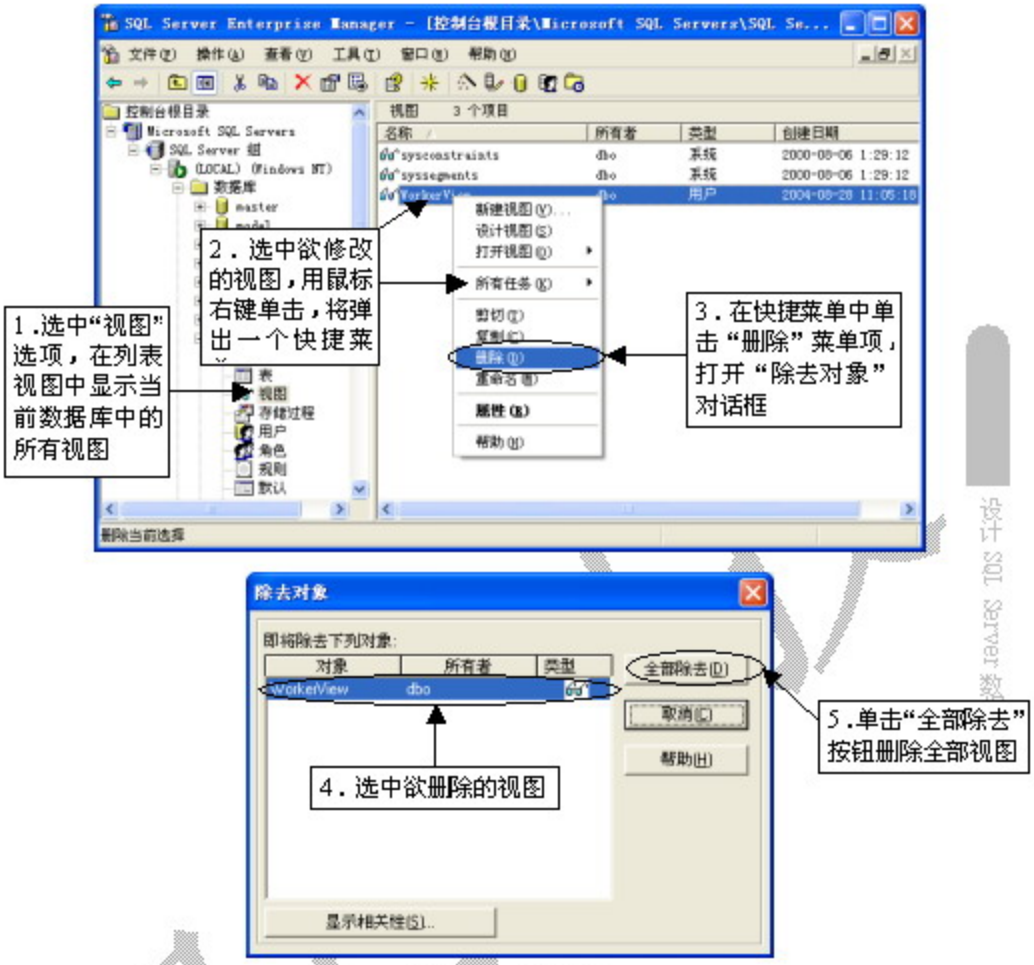

SQL Server Enterprise Manager - [控制台根目录\Microsoft SQL Servers\SQL Se...
文件(F) 操作(A) 查看(V) 工具(T) 窗口(W) 帮助(H)
控制台根目录
Microsoft SQL Servers
SQL Server 组
(LOCAL) (Windows NT)
数据库
master
model
表
视图
存储过程
用户
角色
规则
默认
视图 3 个项目
名称
所有者
类型
创建日期
sysconstraints dbo 系统 2000-08-06 1:29:12
syssegments dbo 系统 2000-08-06 1:29:12
dbo 用户 2004-09-28 11:05:10
新建视图(V)...
设计视图(E)
打开视图(O)
所有任务(K)
剪切(T)
复制(C)
删除(D)
重命名(M)
属性(R)
帮助(H)
删除当前选择
1.选中“视图”选项，在列表视图中显示当前数据库中的所有视图
2．选中欲修改的视图，用鼠标右键单击，将弹出一个快捷菜
3．在快捷菜单中单击“删除”菜单项，打开“除去对象”对话框
除去对象
即将除去下列对象:
对象
所有者
类型
WorkerView dbo
全部除去(D)
取消(C)
帮助(H)
显示相关性(S)...
4．选中欲删除的视图
5.单击“全部除去”按钮删除全部视图

图 11.30　删除视图

编辑 SQL Server 数据库

第 12 章

12.1 录入和修改数据

12.1.1 添加新记录

在创建完数据表之后，需要向数据表中录入数据。可以通过企业管理器直接向数据表中录入数据，方法如下。

（1）打开企业管理器，展开“数据库”选项，选择表所在的数据库，如图 12.1 所示。

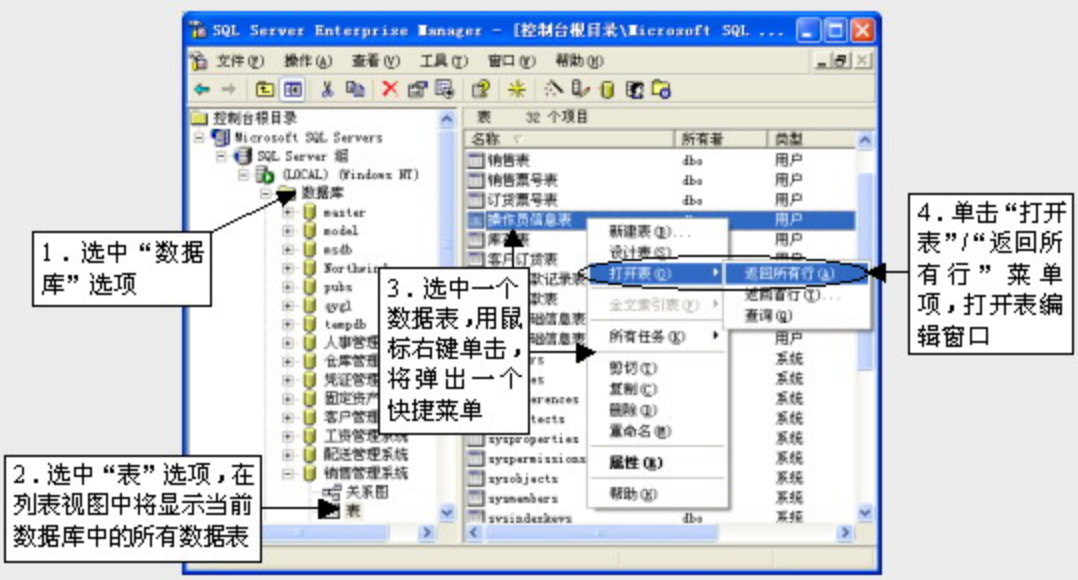


图 12.1 打开表编辑窗口

（2）在表编辑窗口中录入数据，如图 12.2 所示。

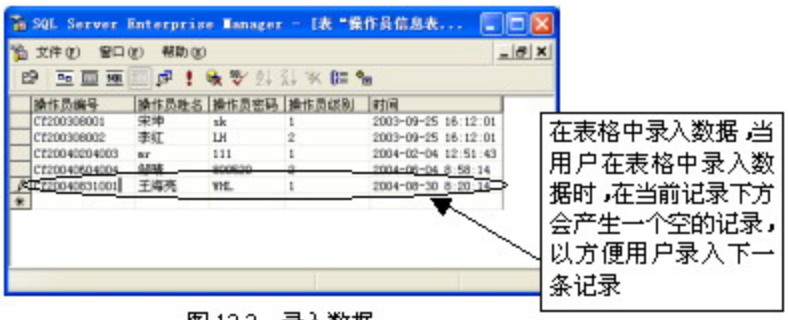


图 12.2　录入数据

注意：在输入数据时，除了数据类型要符合字段类型外，数据还要符合表的各种约束。

12.1.2　记录修改

打开企业管理器，选中欲修改的数据表，打开其表编辑窗口，如图 12.3 所示。



图 12.3　修改数据

注意：在修改数据时，数据类型要符合字段类型，数据也要符合表的各种约束。

12.1.3　删除记录

打开企业管理器，展开“数据库”选项，选中欲删除记录的数据表，打开表编辑窗口，如图 12.4 所示。

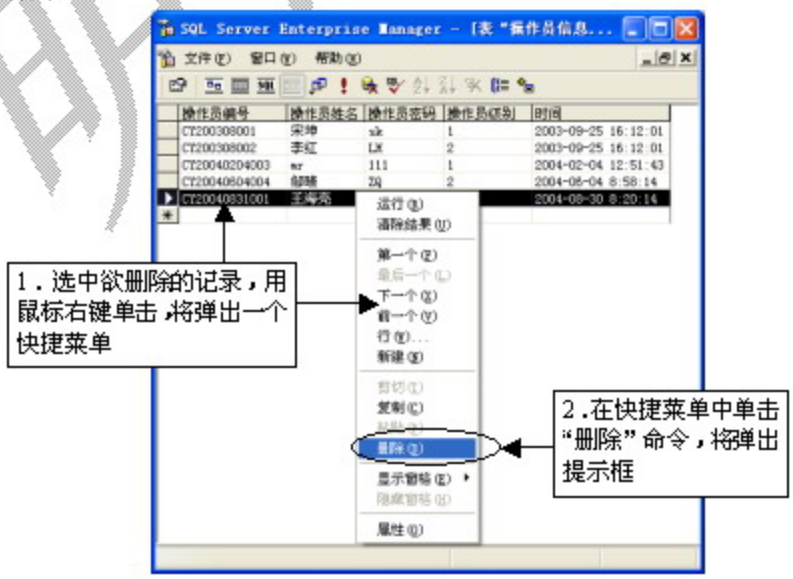




图 12.4　删除记录

注意：如果数据表与其他表有关联，可能不允许删除数据，或者进行级联删除，将其他数据表中的相关数据也删除。

12.2　浏览数据

打开企业管理器，选择数据库和表，将光标移到要浏览的数据表，单击鼠标右键，将弹出快捷菜单，在快捷菜单中选择“打开表”/“返回所有行”菜单项，将打开“库存信息表”编辑窗口，即可浏览数据，如图 12.5 所示。

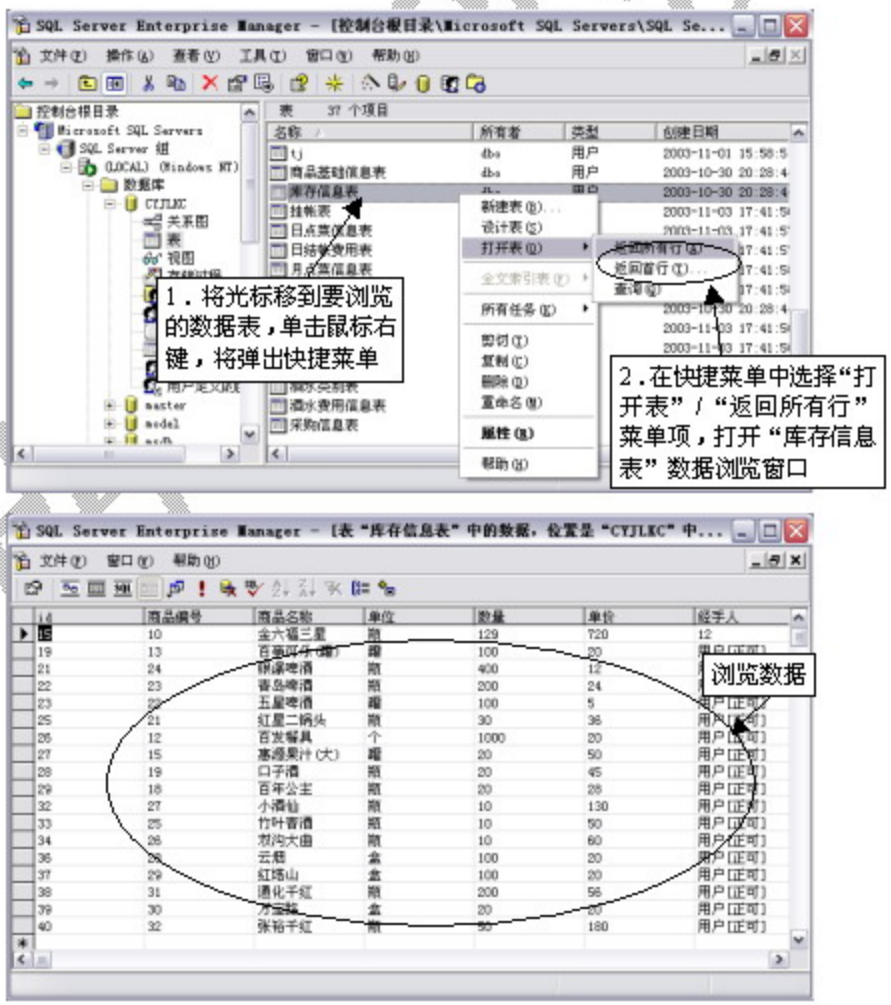


图 12.5　浏览数据

12.3 查找数据

12.3.1 查找指定行记录

打开企业管理器，选中欲定位行的数据表，打开其表编辑窗口，具体操作如图 12.6 所示。

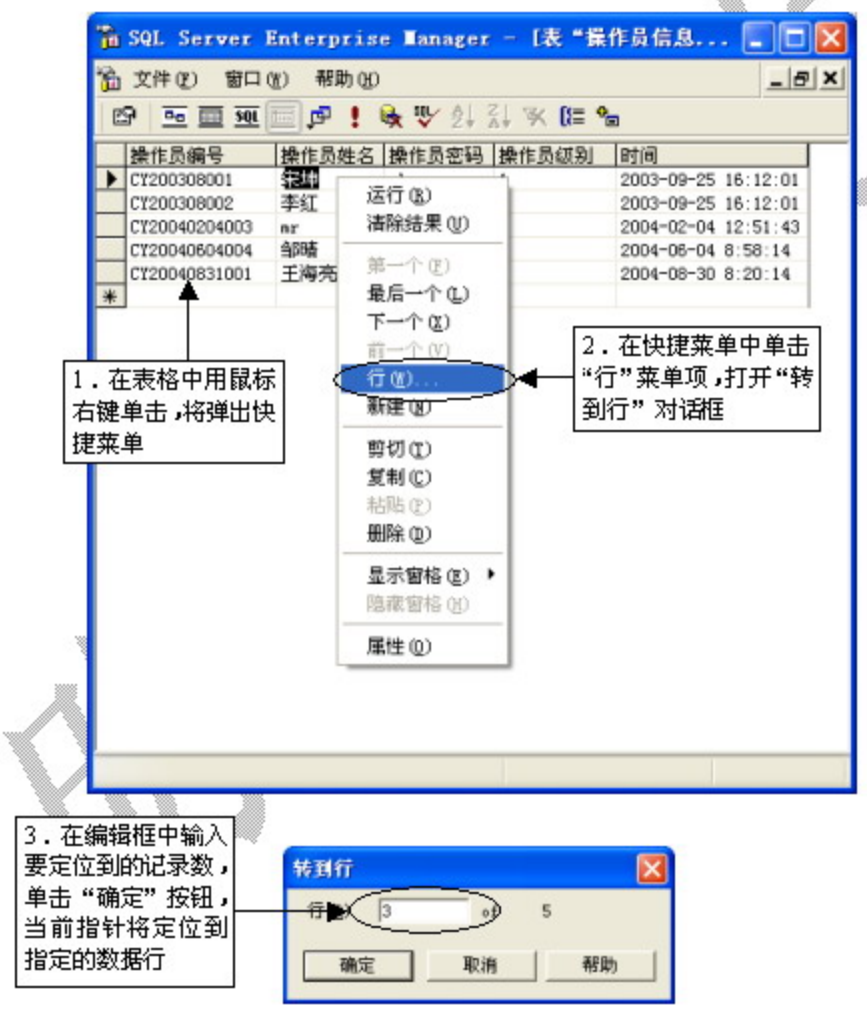


图 12.6 定位数据行

在编辑框中输入要定位到的行，单击"确定"按钮，当前指针会定位到相应的数据行，如果输入的数值大于表格的记录数，当前指针会定位到最后一行。

12.3.2 查找第一条记录

打开企业管理器，选中欲定位行的数据表，打开其表编辑窗口，具体操作如图 12.7 所示。

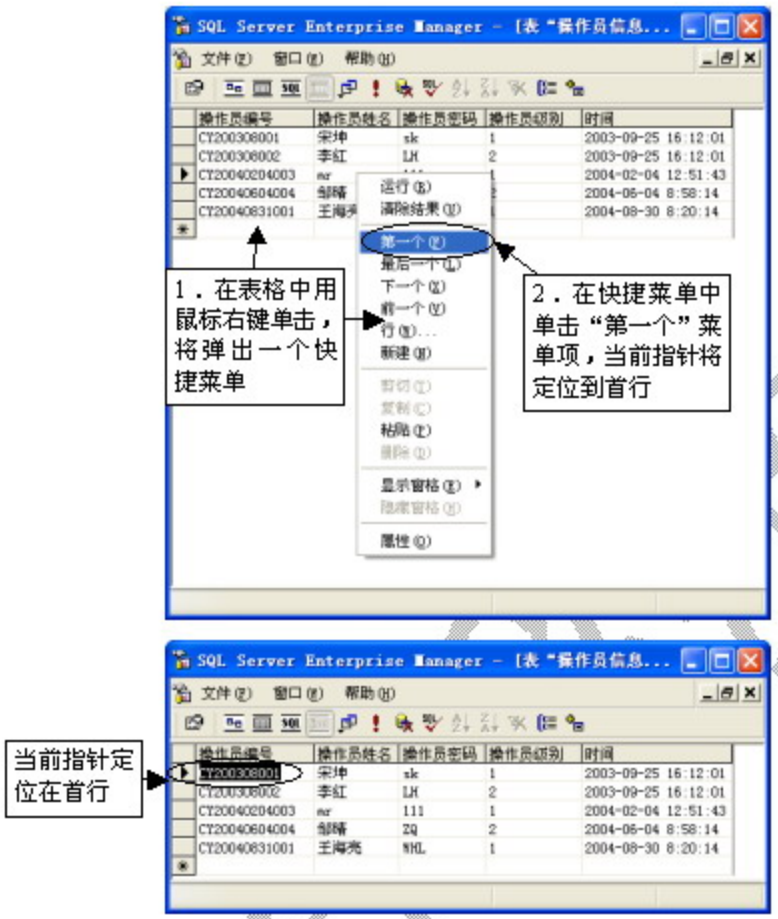
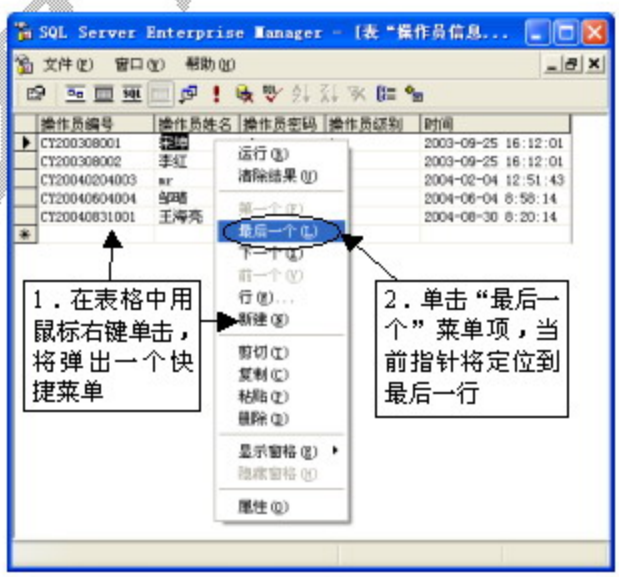


图 12.7　定位首行

12.3.3　查找最后一条记录

打开企业管理器，选中欲定位行的数据表，打开其表编辑窗口，具体操作如图 12.8 所示。



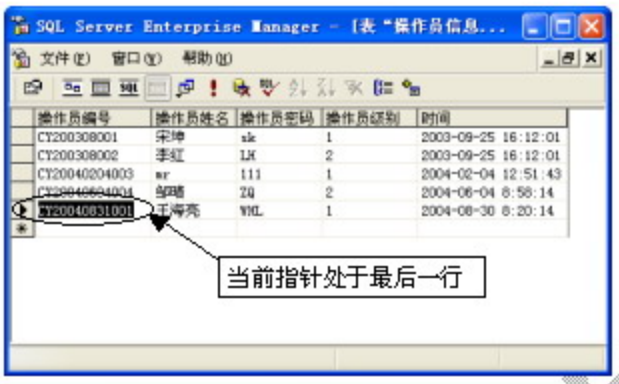


图 12.8　定位最后一行

12.3.4　查找下一条记录

打开企业管理器，选中欲定位行的数据表，打开其表编辑窗口，具体操作如图 12.9 所示。

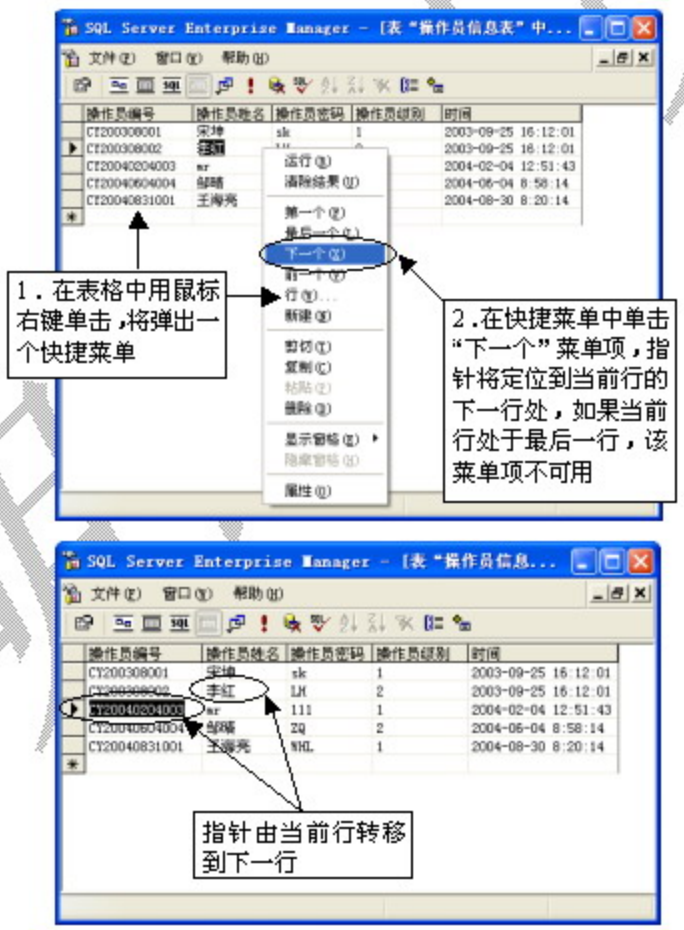


图 12.9　定位下一行

12.3.5　查找前一条记录

打开企业管理器，选中欲定位行的数据表，打开其表编辑窗口，具体操作如图 12.10 所示。

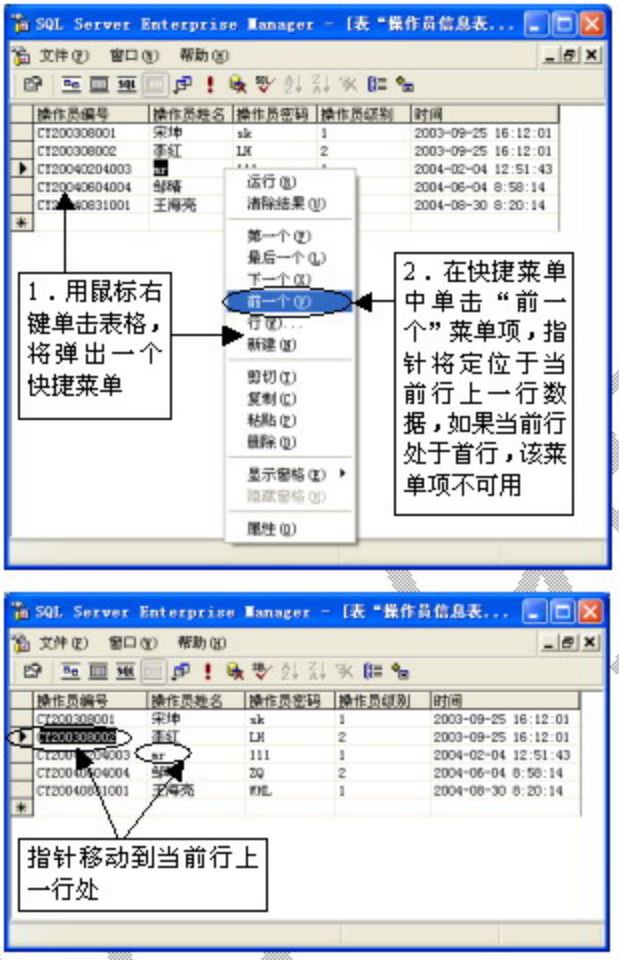


图 12.10　定位前一条记录

第 13 章 SQL 语句

13.1 创建查询和测试查询

13.1.1 编写 SQL 语句

SQL（Structured Query Language）是一种组织、管理和检索存储在数据库中数据的工具，它是一种计算机语言，可以与数据库交互。在 SQL Server 2000 中，用户可以在查询分析器中编写 SQL 语句来操作数据库。打开 SQL Server 2000 的查询分析器，如图 13.1 所示。

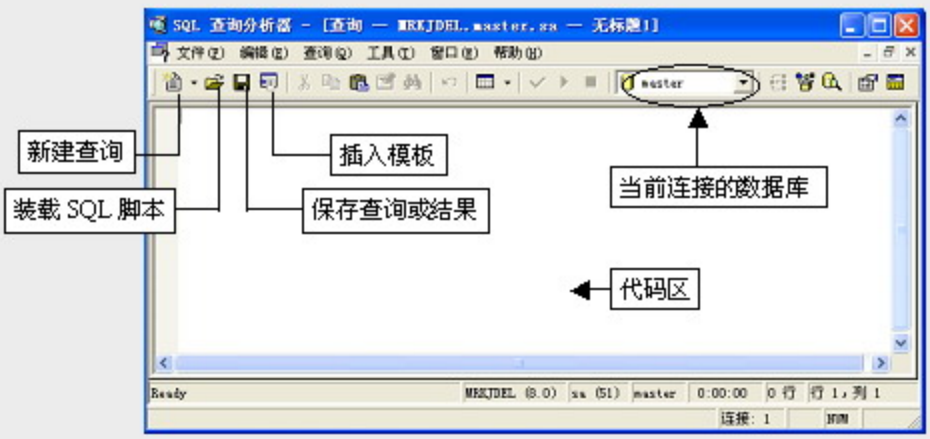


图 13.1 查询分析器

在工具栏的下拉列表框中显示了当前连接的数据库，用户可以在列表框中选择数据库来改变当前的连接。查询分析器中的空白区域是代码区，用户可以在代码区输入 SQL 语句。

13.1.2 测试 SQL 语句

在查询分析器的代码区输入完 SQL 语句后，为了查看语句是否有语法错误，需要对 SQL 语句进行测试。单击工具栏中的按钮或直接按 Ctrl+F5 键可以对当前的 SQL 语句进行测试，如果 SQL 语句有错误，会在代码区下方出现错误提示信息。

13.1.3 执行 SQL 语句

在查询分析器的代码区输入完 SQL 语句后，需要执行 SQL 语句才能实现各种操作。单击工具栏中的按钮或直接按 F5 键可以执行 SQL 语句，如果 SQL 语句执行失败，会出现错误提示信息。

13.2 选择查询

13.2.1 简单的 Select 查询

Select语句是常用的 SQL 语句之一，通常用来查询数据。它从数据库中检索数据并将数据以结果集的形式显示给用户。打开 SQL Server 2000 的查询分析器，在工具栏的数据库下拉列表中选择当前欲连接的数据库，例如“Pubs”数据库，在代码编辑区输入如下 SQL 语句：

```
Select * From 学生表
```

单击工具栏中的“执行查询”按钮或直接按 F5 键执行 SQL 语句，在代码区下方会显示查询语句返回的结果集，如图 13.2 所示。

学生编号	学生姓名	语文	数学	英语	政治	历史	班级编号
S001	杨振坤	89	75	85	86	96	C9803
S002	刘得庄	80	71	92	76	84	C9803
S003	宋坤	80	86	95	86	92	C9803
S004	赵清伍	75	56	87	59	84	C9803
S005	王海亮	90	85	78	71	77	C9803
S006	刘先勇	86	80	72	93	58	C9803
S007	曲学松	76	89	91	74	87	C9803
S008	吕宪涛	87	67	94	56	79	C9803
S009	夏季雨	80	75	86	74	65	C9803

图 13.2 查询结果

分析上面的 SQL 语句，“Select”是关键字，表示该语句是一条查询语句。“*”表示返回数据表中所有字段中的数据。“From”是关键字，指定数据的来源，即从哪个表中返回数据，其后是数据表名称（例如本例中的“学生表”）。

13.2.2 选择查询字段

用户在查询数据时，有时只需查询数据表中某些字段的数据而不想查询所有的字段。此时，可以将 Select 语句写成如下形式：

```
Select  字段名称[,字段名称...] From 表名称
```

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中

输入如下语句：

```
Select  学生编号,学生姓名,语文,数学  From  学生表
```

按 F5 键执行 SQL 语句，会发现结果集只返回 Select 语句中列出的字段，如图 13.3 所示。



图 13.3　选择字段查询

13.2.3　利用 Where 参数过滤数据

在 Select 语句中使用 Where 关键字可以设置过滤条件，格式如下：

```
Select  字段名称[,字段名称...] From 表名称 Where 查找条件
```

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  *  From  学生表  Where  语文>=80
```

按 F5 键执行 SQL 语句，会发现结果集只返回符合条件的数据，如图 13.4 所示。



图 13.4　条件查询

13.2.4　对查询结果进行排序

使用 Select 语句可以对查询结果进行排序，格式如下：

```
Select  字段名称[,字段名称...] From 表名称  Order By 排序表达式 [ASC|DESC]
```

其中排序表达式指定要排序的列。ASC 表示按升序排列，DESC 表示按降序排列。默认情况下按升序排列。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  *  From  学生表  Order By 语文 Desc
```

按 F5 键执行 SQL 语句，会发现“语文”字段按降序排列，如图 13.5 所示。

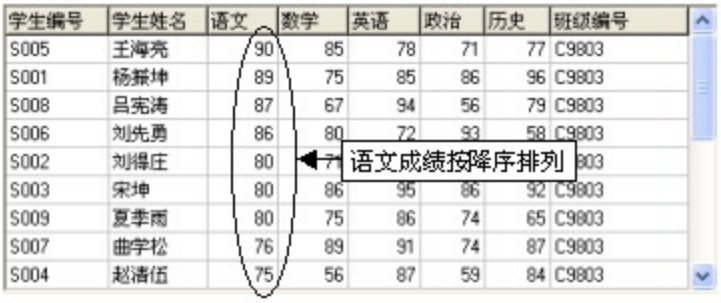


图 13.5　排序查询结果

在进行字段排序时可以同时对多个字段进行排序，SQL 语句如下：

```
Select  *  From  学生表  Order By 语文 Desc,数学 Asc
```

该语句首先会对“语文”字段按降序排列，如果“语文”字段有相同的值，会对这些相同值的数据行中的“数学”字段按升序排列，如果“语文”字段没有相同的值，不对“数学”字段进行排序。

按 F5 键执行 SQL 语句，结果集返回结果如图 13.6 所示。

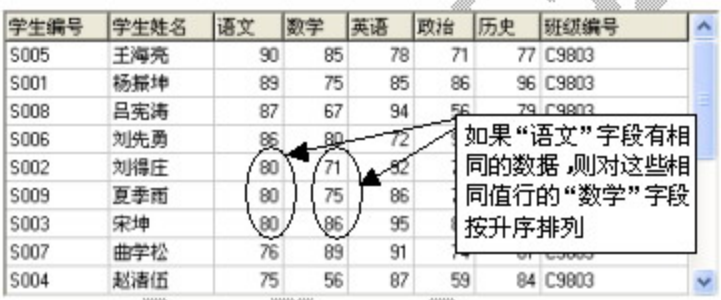


图 13.6　对多个字段进行排序

13.2.5　将查询结果分组统计

1. 语法格式

使用 SQL 语句可以对数据表中的字段进行分组统计，格式如下：

```
Select  聚合函数[聚合函数,...],字段名称[,字段名称...] From 表名称 Group By [All] 统计表达式[,统计表达式...] [With {Cube|Rollup}] [Having 条件表达式]
```

All 是可选项，它包含所有组和结果集，甚至包含那些任何行都不满足 Where 子句指定的搜索条件的组和结果集。如果指定了 All，将对组中不满足搜索条件的汇总列返回空值。

统计表达式是对其分组的表达式，通常情况下为列名。

[With {Cube|Rollup}]是可选项，Cube 指定在结果集内不仅包含由 Group By 提供的正常行，还包含汇总行。在结果集内返回每个可能的组和子组组合的 Group By 汇总行。Rollup 指定在结果集内不仅包含由 Group By 提供的正常行，还包含汇总行。按层次结构顺序，从组内的最低级别到最高级别汇总组。Having 关键字与 Where 关键字类似，用于设置查询条件，只是它用于 Group By 语句中。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select 班级编号,Avg(语文) 语文平均成绩 ,Avg(数学) 数学平均成绩,Avg(英语) 英语平均成绩
From 学生表 Group By 班级编号
```

按 F5 键执行 SQL 语句，返回的结果集如图 13.7 所示。



图 13.7 分组查询

2. 使用 Where 进行条件查询

在进行分组查询时，可以在 Select 语句中指定查询条件，即只对符合条件的记录进行分组统计。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select 班级编号,Avg(语文) 语文平均成绩,Avg(数学) 数学平均成绩,Avg(英语) 英语平均成绩
From 学生表 Where 语文>70 Group By 班级编号
```

按 F5 键执行 SQL 语句，返回的结果集如图 13.8 所示。

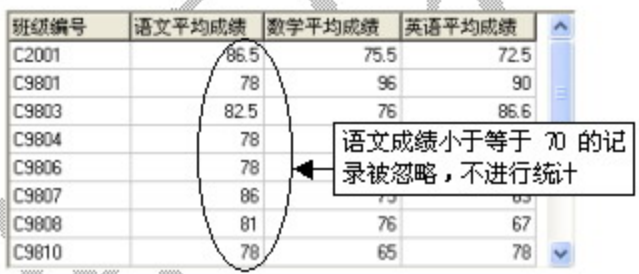


图 13.8 条件分组查询 1

在前面的分组语法格式中介绍了可以在 Group By 语句部分使用 All 关键字，即对于不满足 Where 条件的汇总列返回空值。

在代码区中输入如下语句：

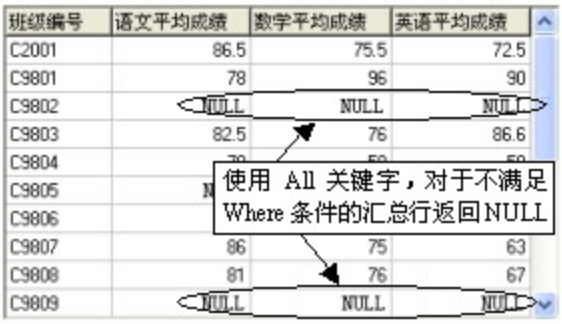
```
Select 班级编号，Avg(语文) 语文平均成绩,Avg(数学) 数学平均成绩，Avg(英语) 英语平均成绩
From 学生表 Where 语文>70 Group By All 班级编号
```

按 F5 键执行 SQL 语句，返回的结果集如图 13.9 所示。

3. 分组搜索条件

在 Group By 语句中可以使用 Having 关键字设置分组条件。打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

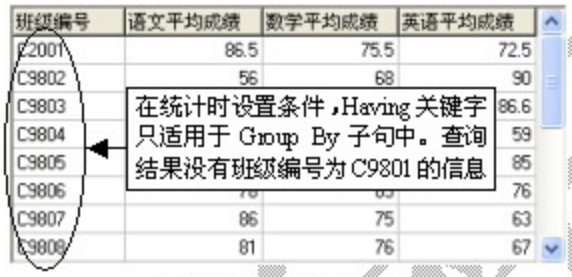
```
Select 班级编号，Avg(语文) 语文平均成绩,Avg(数学) 数学平均成绩，Avg(英语) 英语平均成绩
From 学生表 Group By 班级编号 Having 班级编号<>'C9801'
```



班级编号	语文平均成绩	数学平均成绩	英语平均成绩
C2001	86.5	75.5	72.5
C9801	78	96	90
C9802	NULL	NULL	NULL
C9803	82.5	76	86.6
C9804			
C9805			
C9806			
C9807	86	75	63
C9808	81	76	67
C9809	NULL	NULL	NULL

图 13.9　条件分组查询 2

按 F5 键执行 SQL 语句，返回的结果集如图 13.10 所示。



班级编号	语文平均成绩	数学平均成绩	英语平均成绩
C2001	86.5	75.5	72.5
C9802	56	68	90
C9803			86.6
C9804			59
C9805			85
C9806			76
C9807	86	75	63
C9808	81	76	67

图 13.10　设置分组条件

4. 对多个字段进行分组

SQL 语句可以依据两个或多个字段的内容进行分组。例如，下面的 SQL 语句按照班级和学生姓名将学生成绩分组，它会统计每班的每一名学生的总成绩，语句如下：

```
Select  班级编号,学生姓名,sum(语文+数学+英语+政治+历史) 总成绩 From 学生表
  Group By  班级编号,学生姓名
```

按 F5 键执行 SQL 语句，返回的结果集如图 13.11 所示。

班级编号	学生姓名	总成绩
C9806	付利平	362
C9802	李可平	371
C9804	李平	332
C9803	刘得庄	403
C9807	刘明	343
C9803	刘先勇	389
C9803	吕宪涛	383
C9803	曲学松	417

图 13.11　多个字段分组

5. 使用 Cube 关键字与 Rollup 关键字进行分组

在 Group By 语句中使用 Cube 关键字，在结果集内不仅包含 Group By 语句提供的正常值，还包含汇总行。在结果集内返回每个可能的组和子组组合的 Group By 汇总行。

在查询分析器中输入如下 SQL 语句：

```
Select  班级编号,学生姓名,sum(语文+数学+英语+政治+历史) 总成绩 From 学生表
```

```
Group By  班级编号,学生姓名  with Cube
```

按 F5 键执行 SQL 语句，返回的结果集如图 13.12 所示。

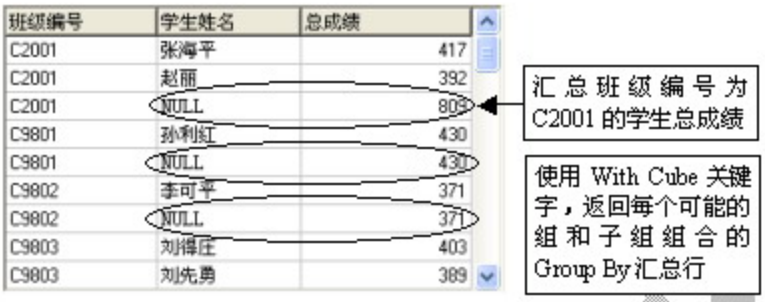


图 13.12　统计所有汇总数据

说明：Cube 关键字指明在结果集内返回每个可能的组与子组组合的 Group By 汇总行，组是指在 Group By 语句中不指明 Cube 关键字所返回的记录。例如图 13.13 中返回的第 1 条记录。子组是指 Group By 语句中指明 Cube 关键字所返回的记录，子组通常在结果集的分组字段中含有空值。例如图 13.12 中返回的第 3 条记录，它表示的是对班级为“C2001”的所有学生记录的统计结果。

在 Group By 语句中使用 Rollup 关键字，在结果集内不仅包含 Group By 语句提供的正常值，还包含汇总行。它按层次结构顺序，从组内的最低级别到最高级别汇总组。

在查询分析器中输入如下 SQL 语句：

```
Select  班级编号,学生姓名,sum(语文+数学+英语+政治+历史) 总成绩 From 学生表
Group By  班级编号,学生姓名  with Rollup
```

按 F5 键执行 SQL 语句，返回的结果集如图 13.13 所示。



图 13.13　汇总组

13.2.6　模糊查询

模糊查询是指 Select 语句返回与指定条件匹配的数据。使用模糊查询需要在 Select 语句中使用关键字 Like，语法格式如下：

```
Select  字段名称[,字段名称…] From 表名称  Where 字段名称 [Not] Like 通配符 表达式
```

在 SQL Server 2000 中通配符共有 4 个，分别为“%”、“_”、“[]”与“[^]”。

“%”表示可以包含零个或多个字符的任意字符串。

“_”代表任意单个字符。

“[]”代表指定范围或集合中的任意一个字符。

“[^]”代表不属于指定范围或集合中的任意一个字符。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  * From 班级表  where 班级编号 like  '%99%'
Select  * From 班级表  where 班级编号 like  '_2001'
Select  * From 班级表  where 班级编号 like  '[a-b]%'
Select  * From 班级表  where 班级编号 like  '[^a-b]%'
```

第一条语句返回处于“班级编号”字段任意位置所包含的字符串“99”的结果集。第二条语句返回“班级编号”字段以“2001”结尾的 5 个字母的结果集。第三条语句返回以字母“a”或字母“b”开头的所有字符串。第四条语句返回不以字母“a”或字母“b”开头的所有字符串。按F5键执行 SQL 语句，结果集如图 13.14 所示。



图 13.14　模糊查询

13.2.7　在查询语句中进行计算

使用 Select 语句不仅可以查询数据表中的数据，而且可以查询由数据表中多个字段组成的计算列。打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select 学生编号,学生姓名,语文+数学+英语 主科成绩 From  学生表
```

按F5键执行 SQL 语句，结果集如图 13.15 所示。



图 13.15　计算列

说明：在上面的 Select 语句中，“语文+数学+英语”表示一个计算列，其数据是由“语文”列、“数学”列和“英语”列的和组成的。在计算列中支持“+”、“-”、“*”、“/”、“%”

5个算术运算符，分别代表加、减、乘、除、取模操作。

13.2.8 为字段起一个别名

使用Select语句查询数据时，可以为查询的字段起一个别名，在SQL Server 2000中为字段建立别名有3种方式，分别为：

```
Select  字段名称 as 别名[,字段名称...] From 表名称
Select  字段名称  别名 [,字段名称...] From 表名称
Select  别名 = 字段名称 [,字段名称...] From 表名称
```

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select 学生编号,学生姓名,语文成绩 = 语文,数学 as 数学成绩,
英语  英语成绩   From  学生表
```

按F5键执行SQL语句，结果集如图13.16所示。

学生编号	学生姓名	语文成绩	数学成绩	英语成绩
S001	杨振坤	89	75	85
S002	刘得庄	80	71	92
S003	宋坤			95
S004	赵清伍			87
S005	王海亮	90	85	78
S006	刘先勇	86	80	72
S007	曲学松	76	89	91
S008	吕宪涛	87	67	94
S009	夏季雨	80	75	86

为字段起的别名

图13.16 字段别名

13.3 使用聚合函数进行查询

13.3.1 数据汇总查询

在Select语句中可以指定Compute子句对数据进行汇总，实际上，使用Compute子句进行数据汇总与使用Group By子句进行数据分组实现的功能相同，只是Compute子句是为了向后兼容而保留的。

使用Compute子句所需信息如下。

- 行聚合函数名称

例如，SUM、AVG、MIN、MAX或COUNT。

- 要对其执行行聚合函数的列
- 可选的By关键字

该关键字可按一列计算指定行聚合。

说明：在Compute子句中使用By关键字时，在Compute子句前需要使用Order By子句对By关键字指定的数据列进行排序。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  学生编号,学生姓名,语文,数学  From  学生表
Compute avg(语文),avg(数学)
```

按 F5 键执行 SQL 语句，结果集如图 13.17 所示。

学生编号	学生姓名	语文	数学
S001	杨振坤	89.0	75.0
S002	刘得庄	80.0	71.0
S003	宋坤	80.0	86.0
S004	赵清伍	75.0	56.0
S005	王海亮	90.0	85.0
S006	刘先勇	86	
S007	曲学松	76	
S008	吕宪涛	87	
avg		avg	
78.300000		75.700000	

使用 Compute 关键字进行汇总

图 13.17　汇总查询

在 Compute 子句中使用 By 关键字进行分组汇总，SQL 语句如下：

```
Select  学生编号,学生姓名,语文,数学  From  学生表
Order by 语文 ASC Compute avg(语文),avg(数学) by 语文
```

按 F5 键执行 SQL 语句，结果集如图 13.18 所示。

学生编号	学生姓名	语文	数学
S014	张海	56.0	78.0
S016	李可平	56.0	68.0
avg		avg	
56.000000		73.000000	
学生编号	学生姓名	语文	数学
S021	孙利艳	59.0	78.0
avg		avg	
59.000000		78.000000	
学生编号	学生姓名	语文	数学
S004	赵清伍	75.0	56.0
avg		avg	
75.000000		56.000000	

图 13.18　分组汇总

13.3.2　数据平均值查询

在 SQL Server 2000 提供的聚合函数中有一个计算平均值的函数 AVG，使用该函数可以返回组中数据的平均值，格式如下：

```
AVG([All| Distinct] 表达式)
```

说明：All 对所有值进行聚合函数运算，All 是默认设置。Distinct 指定 AVG 操作只使用每个值的惟一实例，而不管该值出现了多少次。表达式是精确数字或近似数字数据类型类别的表达式，不允许使用聚合函数和子查询。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  avg(语文) 语文成绩, avg(数学) 数学成绩
From  学生表
```

按 F5 键执行 SQL 语句，结果集如图 13.19 所示。

在 AVG 函数中使用 Distinct 选项，Select 语句如下：

```
Select  avg(Distinct 语文) 语文成绩, avg( Distinct 数学) 数学成绩
From  学生表
```

按 F5 键执行 SQL 语句，结果集如图 13.20 所示。

语文成绩	数学成绩
78.3	75.7

图 13.19　计算平均值 1

语文成绩	数学成绩
77.9091	75.0625

在 Avg 函数中使用 Distinct 关键字忽略重复行

图 13.20　计算平均值 2

13.3.3　数据总记录数查询

在 Select 语句中使用 Count 聚合函数可以返回结果集的行数。Count 函数格式如下：

```
Count({[All|Distinct 表达式]|*})
```

说明：All 对所有的值进行聚合函数运算，它是默认设置。Distinct 指定 Count 返回惟一非空值的数量。表达式其类型是除 Uniqueidentifier、Text、Image 或 NText 之外的任何类型。不允许使用聚合函数和子查询。“*”指定应该计算所有行以返回表中行的总数。Count(*) 不需要任何参数，而且不能与 Distinct 一起使用。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  Count(语文) as 记录数  From 学生表
```

按 F5 键执行 SQL 语句，结果集如图 13.21 所示。

在 Count 函数中使用 Distinct 关键字，SQL 语句如下：

```
Select  Count(Distinct 语文) as 记录数  From  学生表
```

按 F5 键执行 SQL 语句，结果集如图 13.22 所示。

记录数
20

图 13.21　查询记录数 1

记录数
11

在 Count 函数中使用 Distinct 关键字忽略重复行

图 13.22　查询记录数 2

在 Count 函数中使用“*”关键字，SQL 语句如下：

```
Select  Count(*) as 记录数  From 学生表
```

按 F5 键执行 SQL 语句，结果集如图 13.23 所示。

记录数
20

在 Count 函数中使用*取代字段名，在使用*时，不能使用 Distinct 关键字

图 13.23　查询记录数 3

13.3.4 数据最小值查询

在 Select 语句中可以使用聚合函数 Min 返回在某一集合上对数值表达式求得的最小值。打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  Min(语文) 最低成绩，AVG(语文) 平均成绩 From  学生表
```

按 F5 键执行 SQL 语句，结果集如图 13.24 所示。



图 13.24　查询数据最小值

13.3.5 数据最大值查询

用户在 Select 语句中使用 Max 聚合函数可以返回在某一集合上对数值表达式求得的最大值。打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  Max(语文) 最高成绩,AVG(语文) 平均成绩  From 学生表
```

按 F5 键执行 SQL 语句，结果集如图 13.25 所示。

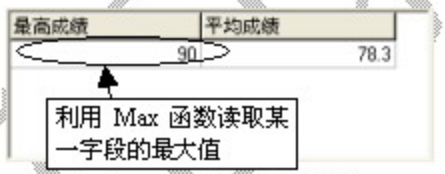


图 13.25　查询数据最大值

13.4 复杂查询

13.4.1 子查询

子查询是在一个查询内的查询。SQL 语句子查询的特征是允许把一个查询的结果作为另一个查询的一部分使用。在 Select 语句中使用子查询需要将子查询语句放在括号内。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  *  From 学生表  where Exists
(Select * from 班级表 where 班级编号 = 'C9803')
```

说明：关键字 Exists 指定一个子查询，检测行的存在。

按 F5 键执行 SQL 语句，结果集如图 13.26 所示。

学生编号	学生姓名	语文	数学	英语	政治	历史	班级编号
S001	杨振坤	89	提示：子查询一定要		86	96	C9803
S002	刘得庄	80	放在括号内		76	84	C9803
S003	宋坤	80	86	95	86	92	C9803
S004	赵清伍	75	56	87	59	84	C9803
S005	王海亮	90	如果子查询有结果		71	77	C9803
S006	刘先勇	86	集返回，将查询学		93	58	C9803
S007	曲学松	76	生表中的所有数据		74	87	C9803

图 13.26　子查询

13.4.2　连接

使用连接，可以根据各个表之间的逻辑关系从两个或多个表中检索数据。当连接表时，创建的连接类型影响出现在结果集内的行，在 SQL Server 2000 中，共有 3 种连接类型，分别如下。

- 内连接

内连接仅显示两个连接表中的匹配行的连接，它是查询设计器中的默认连接类型。

- 外连接

外连接会返回 From 子句中提到的至少一个表或视图的所有行，只要这些行符合任何 Where 或 Having 搜索条件。

- 交叉连接

在这类连接的结果集内，两个表中每两个可能成对的行占一行，如果交叉连接没有使用 Where 子句，结果集是产生连接所涉及的表的笛卡尔积。

13.4.3　内连接

内连接是用比较运算符比较要连接列的值的连接，内连接又分为等连接与不等连接。

1. 等连接

等连接返回在连接列中具有相等值的行，换句话说，在连接条件中使用比较运算符“=”的内连接称为等连接。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select  *  From 班级表 Class  inner join 学生表 Student on
Class.班级编号 = Student.班级编号
```

说明：inner 关键字表示连接类型是内连接，join 关键字表示建立连接，on 关键字用来设置连接条件。该语句的作用是返回“班级表”与“学生表”中所有列，但只返回“班级编号”字段具有相同数据的行。

按 F5 键执行 SQL 语句，结果集如图 13.27 所示。

2. 不等连接

使用等号以外的运算符的内连接称为不等连接。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select Distinct Class1.*  From 班级表 Class1  inner join 班级表 Class2 on
Class1.班级编号 <> Class2.班级编号 and Class1.人数= Class2.人数
```



班级编号	班级名称	班主任	人数	系别	学生编号	学生姓名
C9803	九八级三班	王鹏	68	英语系	S009	夏季雨
C2001	2000级一班	孙军	57	数学系	S012	赵丽
C2001	2000级一班	孙军			S013	张海平
C9805	九八级五班	孙红梅			S014	张海
C9801	九八级一班	白哈森			S015	孙利红
C9802	九八级二班	刘红鹏	42	美术系	S016	李可平
C9804	九八级四班	刘丽红	58	英语系	S017	李平
C9806	九八级六班	张男	61	体育系	S018	付利平

图 13.27　等连接

说明：Distinct 关键字可从 Select 语句的结果中除去重复的行。该语句使用的两个连接表实际上是同一个表，只是为表起了不同的别名，这种连接也称为自连接。该语句的作用是返回“班级表”中人数相同但班级不同的班级信息。

按 F5 键执行 SQL 语句，结果集如图 13.28 所示。



班级编号	班级名称	班主任	人数	系别
C9801	九八级一班	白哈森	68	数学系
C9803	九八级三班	王鹏	68	英语系

图 13.28　不等连接

13.4.4　外连接

内连接只有在同属于两表的行符合连接条件时才返回行，而外连接会返回 From 子句中提到的至少一个表或视图的所有行，只要这些行符合任何 Where 或 Having 搜索条件。在 SQL Server 2000 中外连接有 3 种类型：左向外连接、右向外连接、完整外部连接。

1. 左向外连接

左向外连接又称左连接，结果集会返回左表中所有符合 Where 或 Having 搜索条件的记录。打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select Class.班级编号,Class.班级名称,Class.班主任,
Student.学生编号,Student.学生姓名 From 班级表 Class Left Outer Join
学生表 Student On Class.班级编号 = Student.班级编号
```

说明：Left Outer 关键字表示连接类型是左向外连接，其中 Outer 可以省略。该语句会返回“班级表”中所有的班级信息，而不是只返回与“学生表”中具有相同班级编号的班级信息。

按 F5 键执行 SQL 语句，结果集如图 13.29 所示。

2. 右向外连接

右向外连接又称右连接，结果集会返回右表中所有符合 Where 或 Having 搜索条件的记录。打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select Class.班级编号,Class.班级名称,Class.班主任,
Student.学生编号,Student.学生姓名 From 学生表 Student Right Outer Join
班级表 Class On Class.班级编号 = Student.班级编号
```



班级编号	班级名称	班主任	学生编号	学生姓名
A9901	九九级一班	王可南	NULL	NULL
A9902	九九级二班	刘红	NULL	NULL
C2001	2000级一班	孙军	S012	赵丽
C2001	2000级一班			
C9801	九八级一班			
C9802	九八级二班			
C9803	九八级三班	王鹃	S001	杨振坤
C9803	九八级三班	王鹃	S002	刘得庄
C9803	九八级三班	王鹃	S003	宋坤

图 13.29　左向外连接

说明：Right Outer 关键字表示连接类型是右向外连接，其中 Outer 可以省略。该语句会返回右表（班级表）中所有的信息，而不是只返回符合查询条件的信息。

按 F5 键执行 SQL 语句，结果集如图 13.30 所示。



班级编号	班级名称	班主任	学生编号	学生姓名
A9901	九九级一班	王可南	NULL	NULL
A9902	九九级二班	刘红	NULL	NULL
C2001	2000级一班	孙军	S012	赵丽
C2001	2000级一班	孙军	S013	张海平
C9801	九八级一班			
C9802	九八级二班			
C9803	九八级三班			

图 13.30　右向外连接

3. 完整外部连接

完整外部连接又称完整连接，结果集会返回两个表中所有符合 Where 或 Having 搜索条件的记录，可以将完整外部连接认为是左向外连接与右向外连接的组合。打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
Select Class.班级编号,Class.班级名称,Class.班主任,
Student.学生编号,Student.学生姓名 From 学生表 Student Full Outer Join
班级表 Class On Class.班级编号 = Student.班级编号
```

说明：Full Outer 关键字表示连接类型是完整外部连接，其中 Outer 可以省略。该语句会返回左表（学生表）与右表（教师表）中所有的记录。

按 F5 键执行 SQL 语句，结果集如图 13.31 所示。



班级编号	班级名称	班主任	学生编号	学生姓名
A9901	九九级一班	王可南	NULL	NULL
A9902	九九级二班	刘红	NULL	NULL
C2001	2000级一班	孙军	S012	赵丽
C2001	2000级一班	孙军	S013	张海平
C9801	九八级一班			
C9802	九八级二班			
C9803	九八级三班			
C9803	九八级三班			

图 13.31　完整外部连接

13.4.5 交叉连接

没有 Where 子句的交叉连接将产生连接所涉及的表的笛卡尔积，结果集的行数等于第一个数据表的行数与第二个数据表的行数的乘积，如果在交叉连接中使用 Where 子句，交叉连接将与内连接一样。

打开查询分析器，在工具栏的数据库下拉列表框中选择“Pubs”数据库，在代码区中输入如下语句：

```
SELECT Class.班级编号,Class.班级名称,Class.班主任,Student.学生编号,Student.学生姓名
FROM 学生表 Student CROSS JOIN 班级表 Class
```

说明：Cross Join 关键字表示连接类型是交叉连接。

按 F5 键执行 SQL 语句，结果集如图 13.32 所示。

班级编号	班级名称	班主任	学生编号	学生姓名
A9901	九九级一班	王可南	S001	杨振坤
A9901	九九级一班	王可南	S002	刘得庄
A9901	九九级一班			宋坤
A9901	九九级一班			赵清伍
A9901	九九级一班			王海亮
A9901	九九级一班			刘先勇
A9901	九九级一班			曲学松
A9901	九九级一班			吕宪涛
A9901	九九级一班			夏季雨

完全外部连接，返回两个表的笛卡尔积，如果学生表中有 16 条记录，班级表中有 10 条记录，查询结果将返回 160 条记录

图 13.32 交叉连接 1

在交叉连接中使用 Where 子句，SQL 语句如下：

```
Select Class.班级编号,Class.班级名称,Class.班主任,
Student.学生编号,Student.学生姓名 From 学生表 Student Cross Join
班级表 Class  where Class.班级编号 = Student.班级编号
```

按 F5 键执行 SQL 语句，结果集如图 13.33 所示。

班级编号	班级名称	班主任	学生编号	学生姓名
C9803	九八级三班	王鹏	S006	刘先勇
C9803	九八级三班	王鹏	S007	曲学松
C9803	九八级三班			吕宪涛
C9803	九八级三班			夏季雨
C2001	2000级一班			赵丽
C2001	2000级一班			张海平
C9805	九八级五班	孙红梅	S014	张海
C9801	九八级一班	白哈森	S015	孙利红
C9802	九八级二班	刘红鹃	S016	李可平

在交叉连接中指定 Where 条件，该交叉连接相当于一个内连接

图 13.33 交叉连接 2

13.5 数据添加

13.5.1 Insert 语句介绍

在 SQL 语句中提供了 Insert 语句进行数据添加，其语法格式如下：

```
INSERT [Into] table_name [(column_list)] values (data_values)
```

说明：INSERT 是关键字，Into 是可选项，table_name 为表名，即向哪个表中插入数据。column_list 为表中的列名，如果为多列，则列名间用逗号隔开，省略该参数，即向表中所有列插入数据。data_values 为向表中插入的数据。

13.5.2 Insert 语句基本应用

向“操作员信息表”中添加操作员，具体操作如图 13.34 所示。

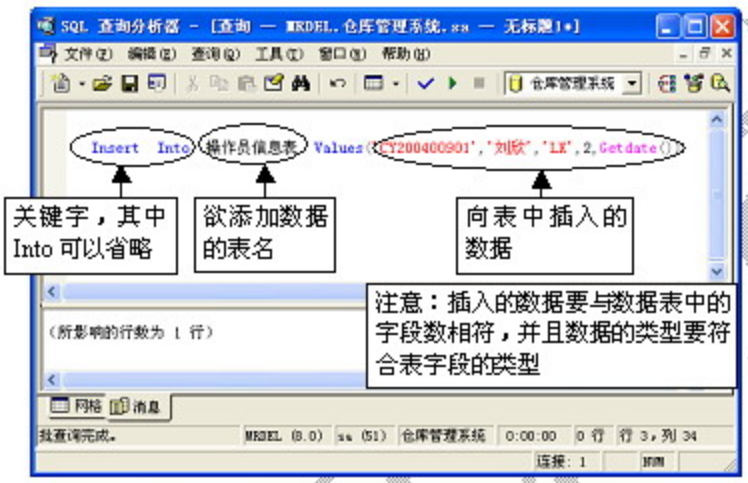


图 13.34 向表中插入数据

13.6 数据修改

13.6.1 Update 语句介绍

在 SQL 语句中提供了 Update 语句对数据进行修改，其语法格式如下：

```
UPDATE    table_name
  SET  { column_name = { expression | DEFAULT | NULL }[, ...n]
[WHERE  search_condition  ]
```

说明：Update 为关键字，table_name 是欲修改的表名，SET 关键字指定修改哪些字段，column_name 是欲修改的字段名，expression 是为字段指定的新值，DEFAULT 表示使用字段的默认值，NULL 表示将字段值设为 Null（空）。WHERE 关键字是可选项，用来设置修改条件，即只对符合条件的记录进行修改，如果不指定 WHERE 条件，会对数据表中的所有记录进行修改。

13.6.2 Update 语句基本应用

修改“库存表”中的信息，设置商品编号为“SP2004001”的商品库存金额为 12 000。如图 13.35 所示。

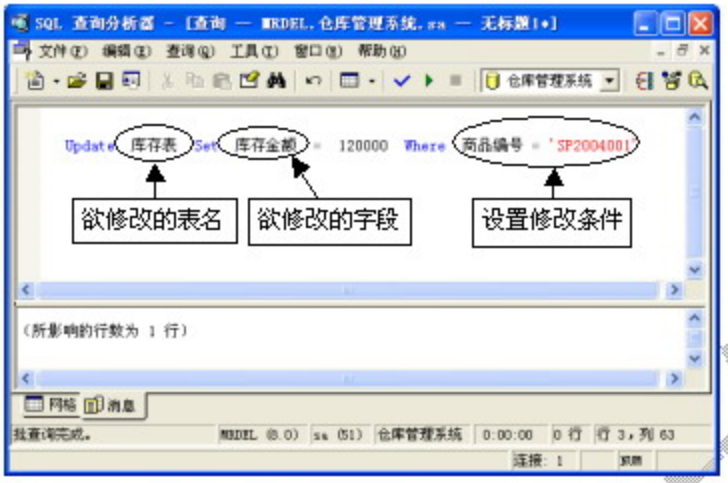


图 13.35　数据修改

13.7　数 据 删 除

13.7.1　Delete 语句介绍

在 SQL 语句中提供了 Delete 语句来删除数据，其语法格式如下：

```
DELETE table_name  [WHERE  search_condition]
```

说明：DELETE 是关键字，table_name 是欲删除的表名，WHERE 是可选项，用来设置删除条件，即只有符合条件的记录才被删除，如果不指定条件，会将表中的所有数据删除。

13.7.2　Delete 语句基本应用

删除“库存表”中商品编号为“SP2004001”的商品信息，如图 13.36 所示。

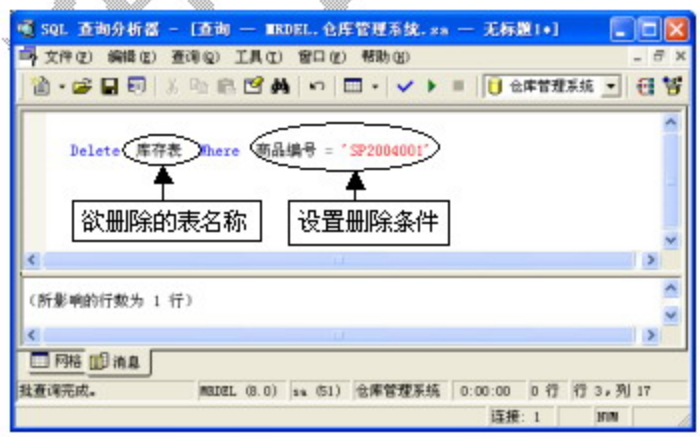


图 13.36　数据删除

第 14 章 存储过程、触发器与视图

14.1 存储过程概述

存储过程是保存起来的可以接收和返回用户提供的参数的 Transact-SQL 语句的集合。在存储过程中可以使用数据存取语句、流程控制语句、错误处理语句等。其主要特点是执行效率高，可重复使用。在创建存储过程时，SQL Server 会将存储过程编译成一个执行计划并保存起来，在执行存储过程时，不需要重新编译，因此执行速度快。一旦创建一个存储过程，很多需要执行该过程的应用程序都可以调用存储过程，减少程序员可能出现的错误。

14.2 存储过程的应用

14.2.1 新建存储过程

在 SQL Server 中可以使用“CREATE PROCEDURE”语句创建存储过程，其语法格式如下：

```
CREATE PROC [EDURE] procedure_name [;number]
  [{@parameter data_type}
    [VARYING] [= default] [OUTPUT]
  ] [,...n]
AS sql_statement
```

说明：CREATE PROCEDURE 是关键字，也可以写成 CREATE PROC。procedure_name 是欲创建的存储过程名称，;number 是可选项，表示对存储过程进行分组，以后可以用一条 DROP PROCEDURE 语句即可将同组的过程一起除去。@parameter 是存储过程参数，存储过程

可以声明一个或多个参数。用户必须在执行过程时提供每个所声明参数的值（除非定义了该参数的默认值）。data_type 是参数的数据类型，所有数据类型（包括 Text、NText 和 Image 等）均可以用做存储过程的参数。不过，cursor 数据类型只能用于 OUTPUT 参数。VARYING 是可选项，指定作为输出参数支持的结果集（由存储过程动态构造，内容可以变化），该关键字仅适用于游标参数。default 是可选项，表示为参数设置默认值。OUTPUT 是可选项，表明参数是返回参数，可以将参数值返回给调用的过程。n 表示可以定义多个参数。AS 关键字指定存储过程要执行的操作。sql_statement 是存储过程中的过程体。

示例如图 14.1 所示。

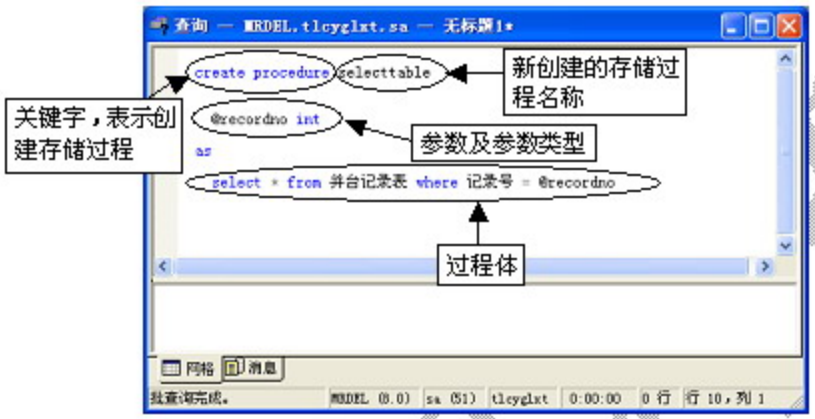


图 14.1　创建存储过程

14.2.2　修改存储过程

在 SQL Server 中，提供了“ALTER PROCEDURE”对存储过程进行修改，其语法格式如下：

```
ALTER PROC [EDURE] procedure_name [;number]
  [{ @parameter data_type }
    [VARYING] [= default] [OUTPUT]
  ] [,...n]
AS sql_statement
```

说明：修改存储过程与创建存储过程类似，只是使用的关键字不同，因此对于 ALTER PROCEDURE 的语法不做详细介绍，请读者参考 14.2.1 小节。

示例如图 14.2 所示。

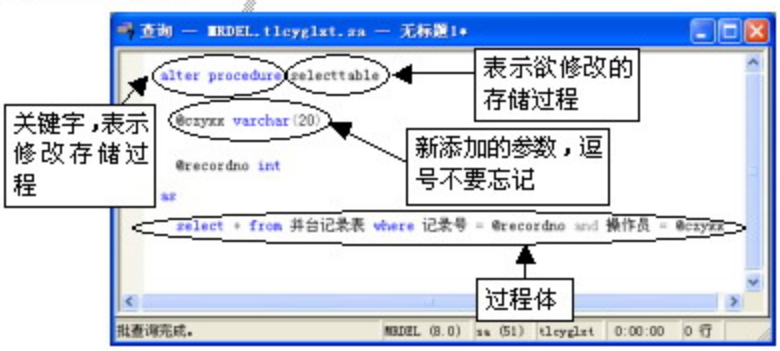


图 14.2　修改存储过程

14.2.3 删除存储过程

在 SQL Server 中，提供了“DROP PROCEDURE”语句用来删除存储过程，其语法格式如下：

```
DROP PROCEDURE { procedure } [,...n]
```

说明：DROP PROCEDURE 是关键字，指定要删除的存储过程，procedure 是要删除的存储过程或存储过程组的名称。n 表示该语句可以同时删除多个存储过程，过程名之间用逗号分隔。

示例如图 14.3 所示。

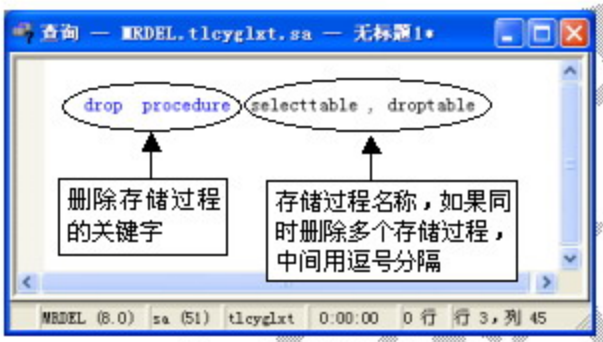


图 14.3 删除存储过程

14.2.4 获取数据库中的存储过程

在 SQL Server 中，表、视图、存储过程等数据库对象信息都记录在系统表“sysobjects”中，因此可以通过该表查看当前数据库中的对象。

示例如图 14.4 所示。

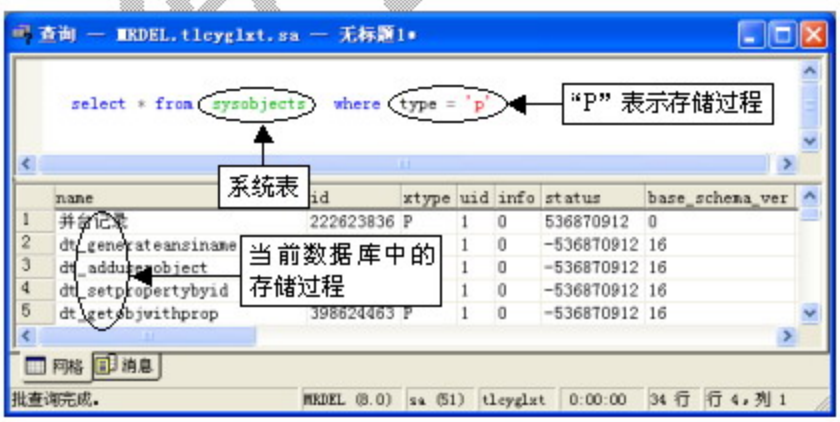


图 14.4 获取存储过程

14.2.5 获取指定存储过程语句

在 SQL Server 中提供了“sp_helptext”系统存储过程来查看规则、默认值、未加密的

存储过程、用户定义函数、触发器或视图的文本，其语法格式如下：

```
sp_helptext [@objname =] 'name'
```

说明：sp_helptext 是关键字，[@objname =] 'name'是对象的名称，对象必须在当前数据库中存在。

示例如图 14.5 所示。

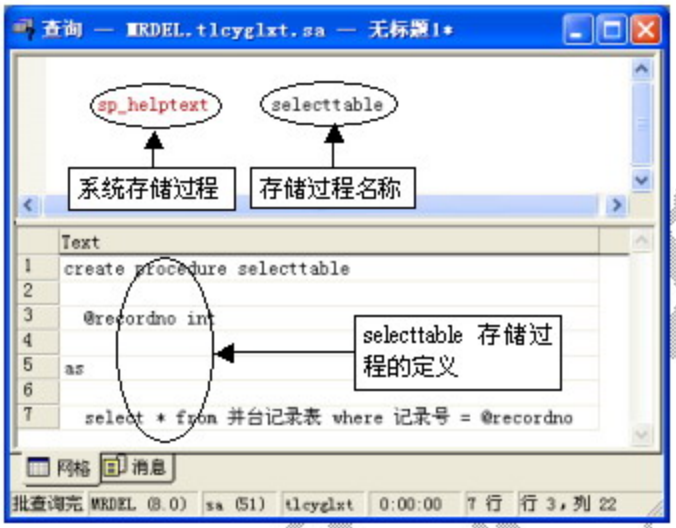


图 14.5 查看存储过程定义

14.2.6 存储过程的调用

在 SQL Server 中，调用存储过程需要使用“EXECUTE”语句，其语法格式如下：

```
[[EXEC[UTE]]
  {
   [@return_status =]
    { procedure_name [;number] | @procedure_name_var
  }
  [[@parameter =] { value | @variable [OUTPUT] | [DEFAULT] }
  [,...n]
```

说明：EXECUTE 是关键字，可以省略，也可以简写为 EXEC。@return_status 是可选项，是一个可选的整型变量，保存存储过程的返回状态，该变量必须在 EXECUTE 语句之前声明过。procedure_name 是存储过程名称，;number 是可选的整数，用于将相同名称的过程进行组合。@procedure_name_var 是局部定义变量名，代表存储过程名称。@parameter 是过程参数，在 CREATE PROCEDURE 语句中定义，参数名称前必须加上符号“@”。value 是过程中参数的值。@variable 是用来保存参数或者返回参数的变量。OUTPUT 指定存储过程必须返回一个参数，使用游标变量作参数时使用该关键字。DEFAULT 根据过程的定义，提供参数的默认值。n 表示 EXECUTE 语句可以指定一个或者多个参数。

示例如图 14.6 所示。

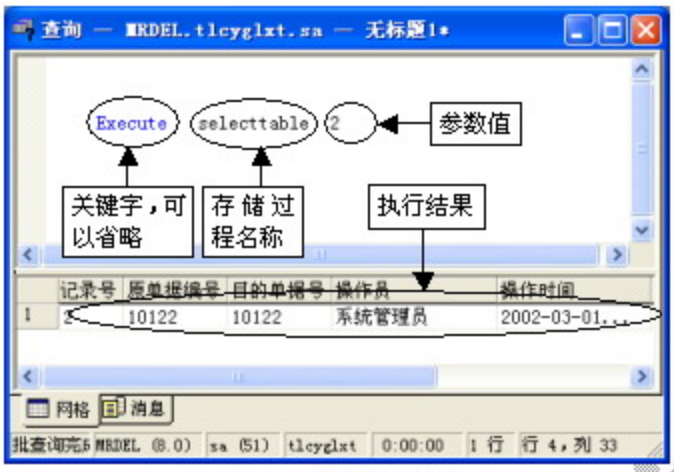


图 14.6　调用存储过程

14.3　触发器介绍

触发器是一种特殊的存储过程，它与数据表结合在一起。当数据表中的数据被更改时，触发器会被触发，执行相应操作，这说明了触发器是由数据库管理系统调用的。

在 SQL Server 2000 中，触发器分为两种，即 After 触发器与 Instead of 触发器。After 触发器是在数据已完成变动后才被激活，执行触发器操作，而 Instead of 触发器会在数据变动前触发，并且它会取代原来进行的操作。

14.4　使用触发器

14.4.1　新建触发器

在 SQL Server 中可以使用“CREATE TRIGGER”语句创建触发器，语法格式如下：

```
CREATE TRIGGER trigger_name
ON { table | view }
{
  { { FOR | AFTER | INSTEAD OF } { [DELETE] [,] [INSERT] [,] [UPDATE] }
  AS
   sql_statement [ ...n]
  }
}
```

说明：CREATE TRIGGER 是关键字，trigger_name 是欲创建的触发器名称。ON 为关键字，table | view 表示在哪个表或视图上执行触发器。FOR 表示触发器为 AFTER 触发器。AFTER 指定触发器只有在触发 SQL 语句中指定的所有操作都已成功执行后才触发，在视图上不能创建 AFTER 触发器。INSTEAD OF 指定执行触发器而不是执行触发 SQL 语句，从而替代触发语句的操作。{ [DELETE] [,] [INSERT] [,] [UPDATE] }指定在表或视图上执行哪些数据修改语句时将激活触发器。AS 表示触发器要执行的操作。sql_statement 是触发器

的条件和操作。n表示触发器可以有多条SQL语句。

示例如图14.7所示。

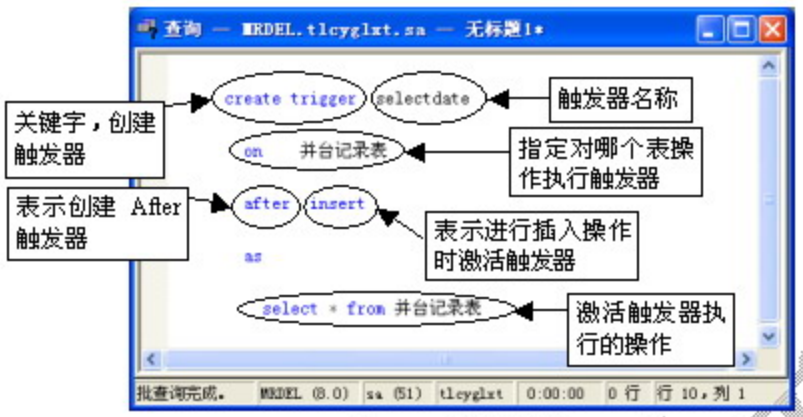


图14.7 创建触发器

14.4.2 修改触发器

在SQL Server中，提供了“ALTER TRIGGER”语句用来修改触发器，其语法格式如下：

```
ALTER TRIGGER trigger_name
ON { table | view }
{
  { { FOR | AFTER | INSTEAD OF } { [DELETE] [,] [INSERT] [,] [UPDATE] }
  AS
   sql_statement [ ...n]
  }
}
```

说明：修改触发器与创建触发器语法格式基本相同，只是关键字不同而已，因此，不再重述，请读者参考14.4.1小节。

示例如图14.8所示。

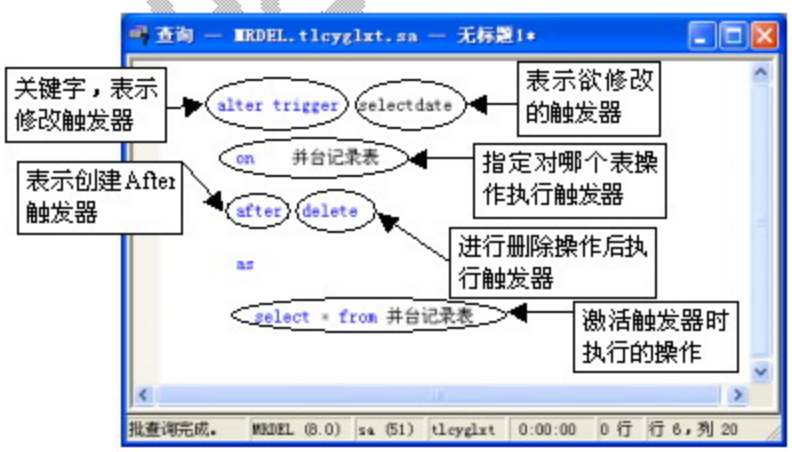


图14.8 修改触发器

14.4.3 删除触发器

在 SQL Server 中删除触发器需要使用“DROP TRIGGER”语句，语法格式如下：

```
DROP TRIGGER { trigger } [,...n]
```

说明：DROP TRIGGER 是关键字，trigger 表示欲删除的触发器名称，n 表示可以同时删除多个触发器，触发器名称间用逗号分隔。

示例如图 14.9 所示。

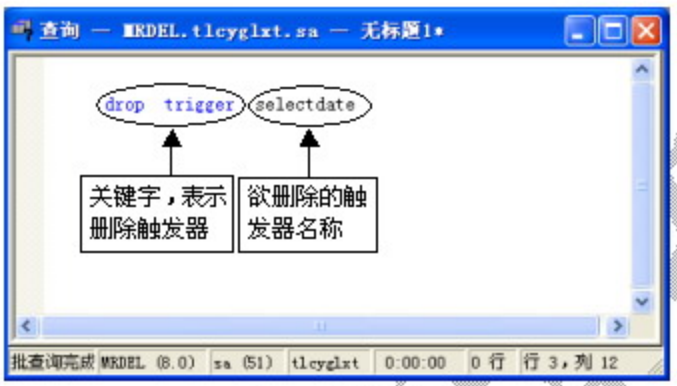


图 14.9 删除触发器

14.5 视图介绍

视图是保存在数据库中的 SELECT 查询，也有人称它为虚拟表。使用视图可以简化查询操作，视图能够从多个数据表中提取数据，并以单个表的形式显示查询结果，这样，可以把针对多表的数据查询转变为对视图的单表查询。在使用视图时也有一些限制。例如，对于简单的视图，可以进行更新操作，但对于复杂的视图，则不允许进行更新操作。

14.6 视图应用

14.6.1 新建视图

在 SQL Server 中，提供了“CREATE VIEW”语句来创建视图，语法格式如下：

```
CREATE VIEW  view_name
AS
select_statement
```

说明：CREATE VIEW 是关键字，表示创建视图，view_name 是新创建的视图名称，AS 表示视图要执行的查询操作。select_statement 表示视图要执行的查询语句。

示例如图 14.10 所示。

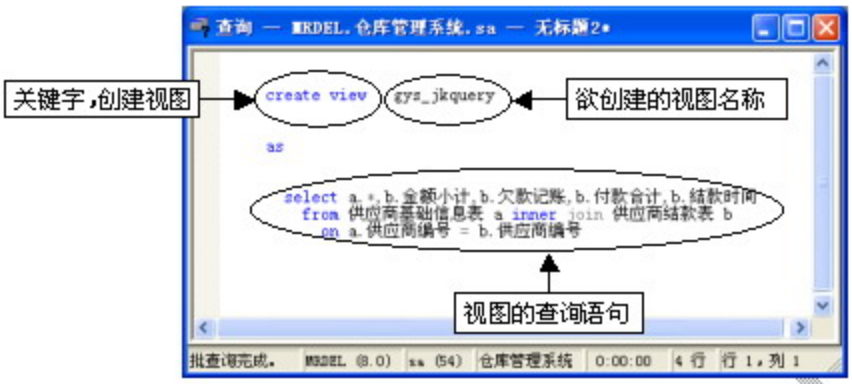


图 14.10　创建视图

14.6.2　修改视图

在 SQL Server 中，修改视图需要使用“ALTER VIEW”语句，语法格式如下：

```
ALTER VIEW   view_name
AS
select_statement
```

说明：修改视图与创建视图的语法格式几乎相同，只是关键字不同，因此不再重述，请读者参考 14.6.1 小节。

示例如图 14.11 所示。

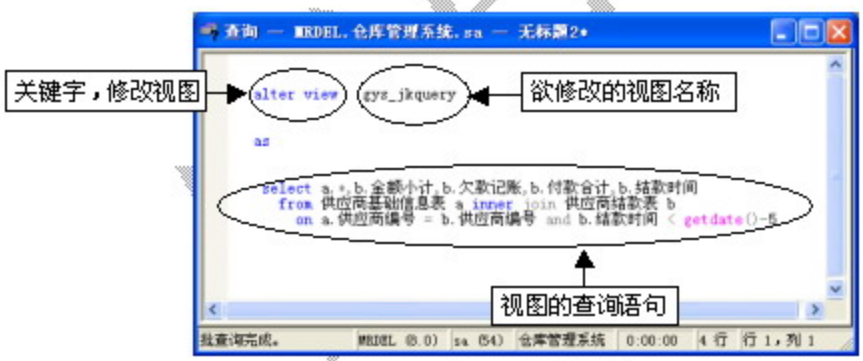


图 14.11　修改视图

14.6.3　删除视图

当不再需要使用视图时，可以使用“DROP VIEW”语句删除视图，语法格式如下：

```
DROP VIEW { view } [,...n]
```

说明：DROP VIEW 是关键字，表示删除视图，view 是欲删除的视图名称，n 表示一次可以删除多个视图，视图名称之间用逗号分隔。

示例如图 14.12 所示。



图 14.12　删除视图

14.6.4　获得数据库中所有的视图

在 14.2.4 小节已经介绍过，SQL Server 中的表、视图、存储过程等数据库对象信息都记录在系统表“sysobjects”中。因此，可以通过查询“sysobjects”表来显示当前数据库中的视图。

示例如图 14.13 所示。

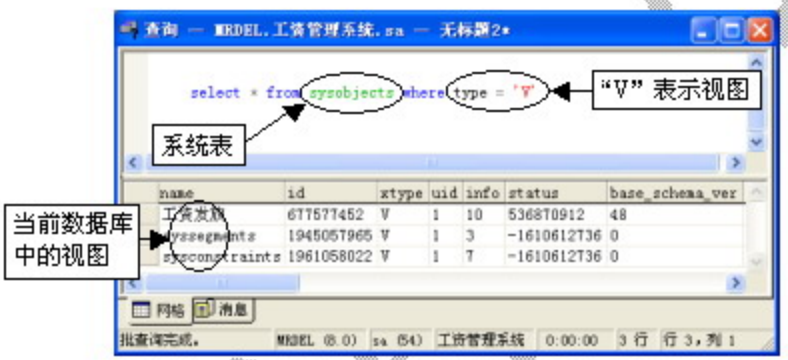


图 14.13　查看视图

14.6.5　获得指定视图语句

在 14.2.5 小节中已经介绍过，SQL Server 提供了“sp_helptext”系统存储过程来查看规则、默认值、未加密的存储过程、用户定义函数、触发器或视图的文本。因此，可以使用“sp_helptext”系统存储过程来查看视图的定义。

示例如图 14.14 所示。

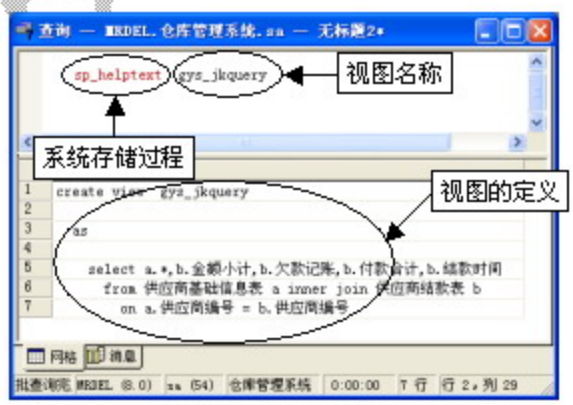


图 14.14　查看视图定义

第 15 章 定制自己的工作环境

在 Delphi 7 的开发环境中包含了许多调试选项和编辑选项，合理地设置这些选项可以提高程序的编译速度，使程序代码更易于阅读和理解，程序开发更加得心应手。

15.1 代码编辑器常规选项设置

单击系统菜单“Tools”/“Editor Options”选项，将弹出编辑器属性对话框，打开“General”选项卡，如图 15.1 所示。

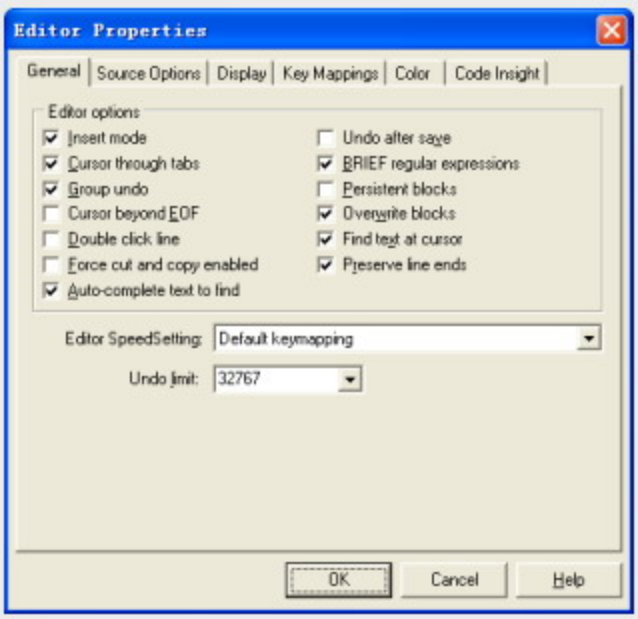


图 15.1 General 选项卡

在“General”选项卡中包含了代码编辑器的行为，各选项功能如下。

- Insert mode

确定代码编辑器是否处于插入状态，如果该选项没有选中，代码编辑器则处于改写状态，此时，在光标处输入的文本会覆盖已存在的文本。可以使用 Insert 键切换代码编辑器的状态，在默认情况下，该选项处于选中状态，建议开发人员不要修改该选项。

- Cursor through tabs

选中该选项，在每个制表符内可以使用方向键移动光标到适当的逻辑位置。

- Group undo

选中该选项，在进行取消操作时，可以取消最后一次一组连续的动作。例如，在代码编辑器中输入“Edit2.Text:='www';”，进行一次取消操作会将该条语句清除。如果该选项没有选中，一次取消操作只能清除一个字符。

- Cursor beyond EOF

选中该选项，按下方向键，光标可以在单元文件的末尾继续向下移动，如果该选项没有选中，按下方向键时，光标在到达单元文件末尾处不会再向下移动，除非按 Enter 键。

- Double click line

选中该选项，当在代码编辑器中双击鼠标时，当前行会被选中。如果该选项处于非选中状态，当在代码编辑器中双击鼠标时，会将临近光标处的字符串选中。

- Force cut and copy enabled

选中该选项，无论代码编辑器中是否有文本被选中，系统菜单“Edit”下的“Cut”和“Copy”菜单项都将可用。

- Auto_complete text to find

选中该选项，在查找对话框中输入要查找的文本时，会将以前查找过的匹配的文本显示在查找对话框中。例如，在查找对话框中输入“Form1”进行查找，当下次再查找数据时，如果在查找对话框中输入以字符“F”开始的字符串时，在查找对话框中会将“Form1”显示出来。

- Undo after save

选中该选项，允许对刚保存过的操作进行撤销。例如，在代码编辑器中输入“Edit2.Text:='Retrieve';”语句，然后进行保存。当进行取消操作时，会发现可以清除该语句。如果该选项没有选中，对刚保存过的数据是不允许撤销的。

- BRIEF regular expressions

选中该选项，在查找对话框中如果选中“regular expressions”选项，可以使用“<”、“<”、“?”等作为统配符查找数据。

- Persistent blocks

选中该选项，如果代码编辑器中有文本被选中，当使用方向键移动光标时，选中标记不会消失，直到有新的文本被选中。如果该选项没有选中，代码编辑器中的文本被选中时，移动光标，选中标记就会消失。

- Overwrite blocks

选中该选项，如果代码编辑器中有文本被选中，当输入字符时，输入的内容会替换被选中的文本，如果“Persistent blocks”选项被选中，输入的字符会插入到当前光标处而不会替换被选中的文本。

- Find text at cursor

选中该选项，当打开查找对话框时，会将当前光标处的文本显示在查找对话框中。如果该选项没有选中，在进行查找数据时，必须自己添加查找内容，除非查找文本列表框为空，如果那样的话，编辑器依然会将当前光标处的文本显示在查找对话框中。

- Preserve line ends

选中该选项，会保存行尾的位置。

- Editor SpeedSetting

使用“Editor SpeedSetting”可以快速地配置编辑器，默认情况下，在“Editor SpeedSetting”下拉列表框中共有 6 个选项，分别对应 6 种不同的配置。

- Undo limit

“Undo limit”标识能够取消的按键次数，默认值为 32767。如果将该选项值设为 2，表示只能连续进行 2 次取消操作。为了查看该选项的作用，可以将“Group undo”选项置为非选中状态，然后在代码编辑器中输入一个字符串，例如“sss”，然后按 Ctrl+Z 组合键进行取消操作，会发现只能进行 2 次取消操作，即清除 2 个字符“s”。

15.2 代码编辑器源文件选项设置

单击系统菜单“Tools”/“Editor Options”选项，将弹出编辑器属性对话框，打开“Source Options”选项卡，如图 15.2 所示。

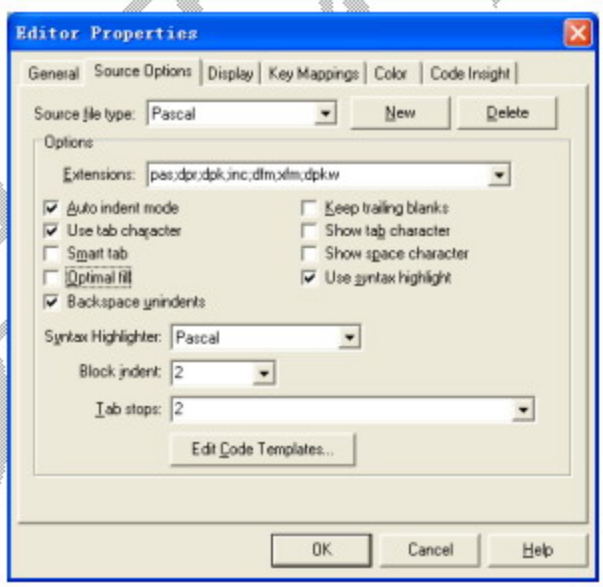


图 15.2 Source Options 选项卡

在“Source Options”选项卡中包含了代码的缩进格式和语法的高亮显示，各选项功能如下。

- Source file type

该选项包含了预先确定的或自定义的源文件类型，当用代码编辑器打开这些源文件时，代码编辑器会根据预先确定的语法高亮显示源文件。

- New button

单击 New 按钮会打开一个新建源文件类型对话框，如图 15.3 所示。

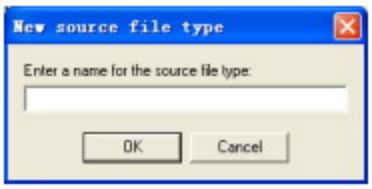


图 15.3　创建源文件类型对话框

如果在编辑框中输入创建源文件类型的名字，并且单击 OK 按钮，无论是否想要创建一个新的源文件类型，都需要在图 15.2 中的“Extensions”下拉列表框中输入文件的扩展名，否则会收到一个错误提示。

❑ Delete button

单击 Delete 按钮会删除一个预定义的或自定义的源文件类型，从源文件类型下拉列表框中选择一个文件类型，单击 Delete 按钮便会将该文件类型删除。

❑ Extensions

该选项标识文件的扩展名，如果在“Extensions”下拉列表框中输入文件的扩展名，它会自动地被添加到下拉列表框中。

❑ Auto indent mode

选中该选项，当在代码编辑器中按 Enter 键时，光标会处在上一行第一个不为空字符的下面。如果该选项没有选中，当在代码编辑器中按 Enter 键时，光标会处在当前行的最前面。

❑ Use tab character

选中该选项，在代码编辑器中按 Tab 键时，会插入制表符，如果该选项没有选中，会用空格符代替制表符。要查看选中该选项的功能，可以选中“Show tab character”选项，然后在代码编辑器中按 Tab 键，效果如图 15.4 所示。

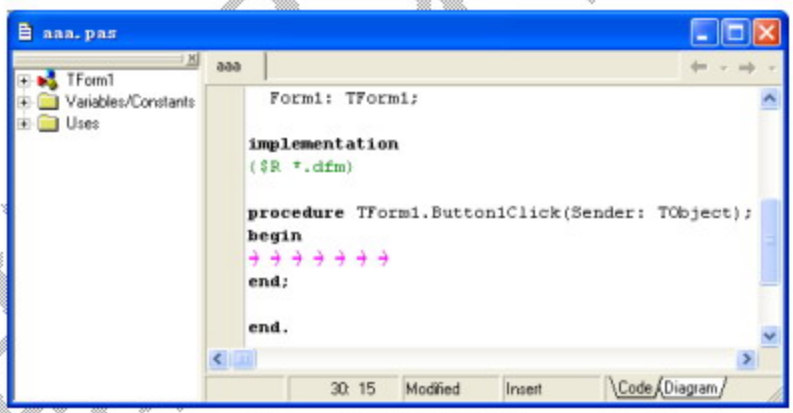


图 15.4　代码编辑器效果图 1

如果“Smart tab”被选中，该选项会处于非选中状态。

❑ Smart tab

选中该选项，如果代码编辑器中当前行第一个不为空的字符在上一行第一个不为空的字符前面（如图 15.5 所示），按 Tab 键会将当前行对齐到上一行第一个不为空的字符处，如图 15.6 所示。

如果“Use tab character”被选中，该选项会处于非选中状态。

❑ Optimal fill

选中该选项，代码编辑器尽量使用最小数量的字符进行行缩进，并且尽量使用制表符代替空格符进行行缩进。要查看该选项的功能，可以选中“Optimal fill”、“Show tab character”、“Use tab character”和“Show space character”选项，如图 15.7 所示。

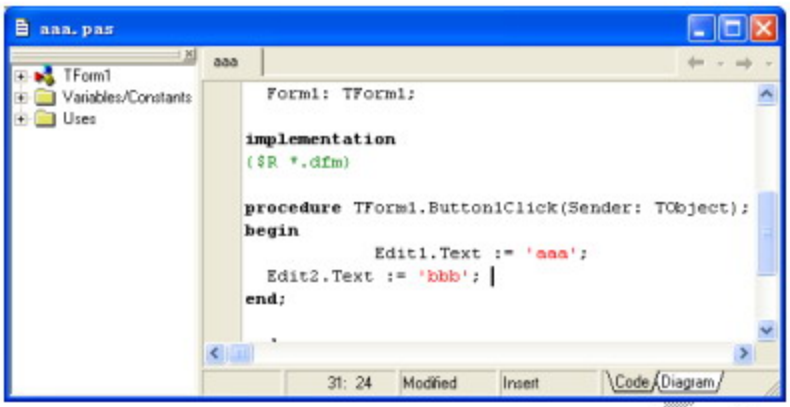


图 15.5　代码编辑器效果图 2

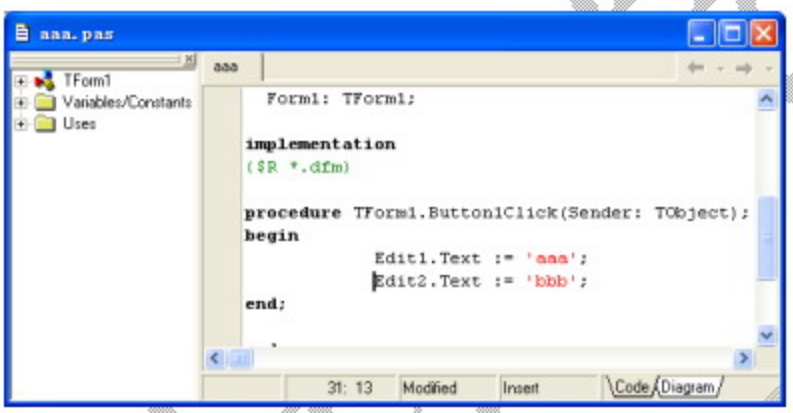


图 15.6　代码编辑器效果图 3

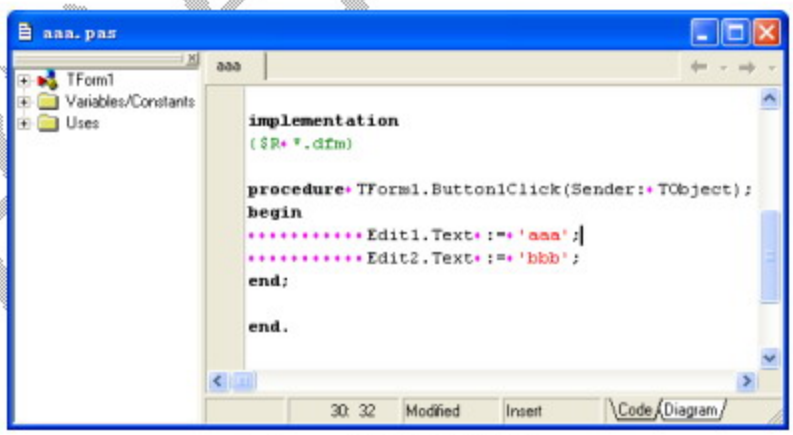


图 15.7　代码编辑器效果图 4

图 15.7 中的一个圆点代表一个空格符。当光标在上、下行移动时，会发现多数空格符被制表符代替了，如图 15.8 所示。

- Backspace unindents

选中该选项，如果光标在当前行第一个不为空字符处，按 BackSpace 键，会将当前行对齐到上一行第一个不为空字符处，前提是上一行第一个不为空字符在当前行第一个不为空字符的前面。如果该选项没有选中，按 BackSpace 键，代码会一个字符一个字符地向前缩进。

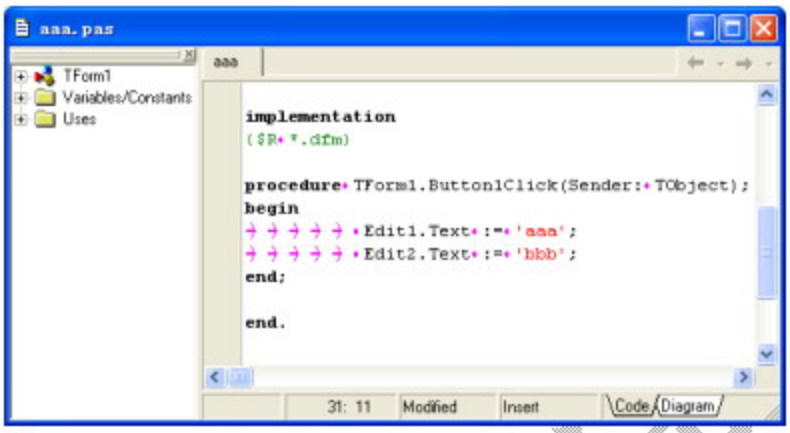


图 15.8　代码编辑器效果图 5

❑　Keep trailing blanks

选中该选项，如果程序代码结尾处有空格，代码编辑器会保留这些空格。如果该选项没有选中，代码编辑器不会保留代码结尾处的空格。要查看该选项的功能，可以选中“Keep trailing blanks”和“Show space character”选项，在程序代码结尾处按空格键，然后在代码编辑器中上、下移动光标，会发现空格依然存在，如图 15.9 所示。

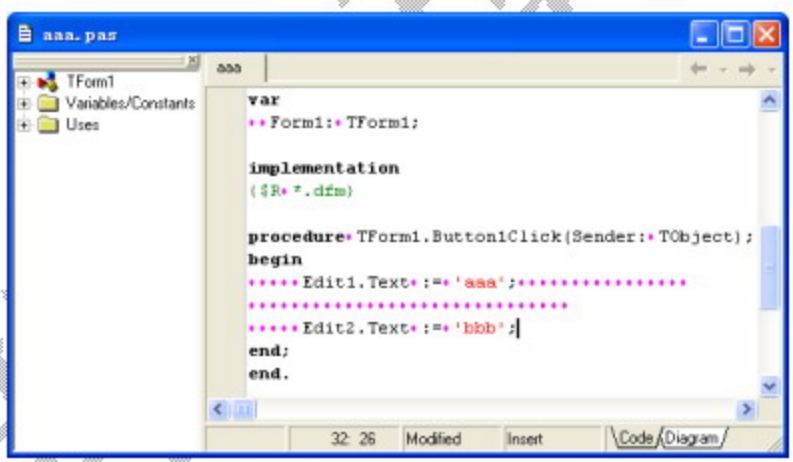


图 15.9　代码编辑器效果图 6

如果该选项没有选中，当上、下移动光标时，代码结尾处的空格会被删除。

❑　Show tab character

该选项用来设置是否在代码编辑器中显示制表符，如果要在代码编辑器中显示制表符，在选中该选项的同时，还需选中“Use tab character”选项。

❑　Show space character

选中该选项，在代码编辑器中会用圆点代表空格符，即一个圆点代表一个空格符。

❑　Use syntax highlight

选中该选项，会将代码编辑器中的程序代码根据“Color”选项卡中的设置显示。

❑　Syntax Highligter

该选项用来选择一种想要特殊显示的语言，如果要使用该选项，首先需要选中“Use syntax highlight”选项。

❑ Block indent

该选项标识缩进的空格数。默认值为 2，上限值为 16，如果在该选项中输入一个大于 16 的数值，会收到一个错误，此种情况在 Windows XP 操作系统中例外，如果在 Windows XP 操作系统下，在该选项中输入一个大于 16 的数值，不会收到错误，并且程序代码会按照指定的数值进行缩进。要查看该选项的功能，可以在代码编辑器中选中一行或几行代码，然后按 Ctrl+Shift+U / I 组合键向前或向后缩进行，会发现代码会按照指定的数值进行缩进。

❑ Tab stops

该选项用来设定按 Tab 键代码缩进的空格数。

❑ Edit Code Templates button

单击【Edit Code Templates】按钮，会打开代码模板对话框，如图 15.10 所示。

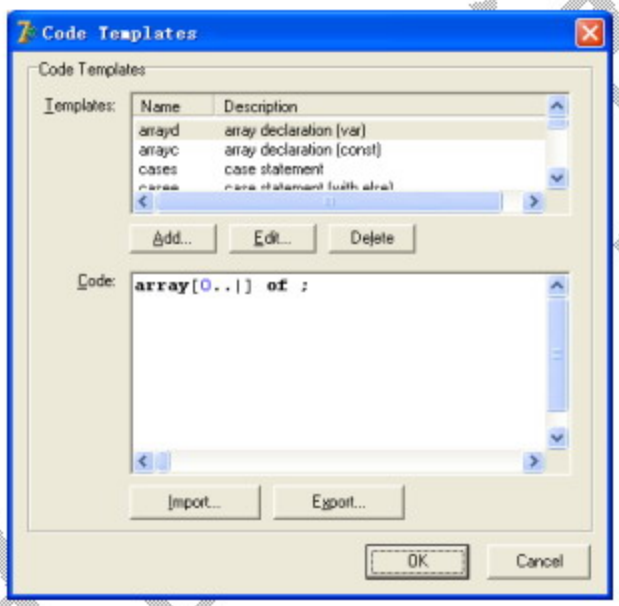


图 15.10　代码模板

在开发程序时，用户经常编写重复的代码，这样不仅麻烦，而且浪费时间。其实，用户可以将经常编写的代码放在代码模板窗口中。当需要编写该代码时，可以在代码编辑器中按〈Ctrl+Shift+J〉组合键打开代码列表对话框，如图 15.11 所示。该对话框会列出当前定义的所有模板，选择你添加的模板，按〈Enter〉键，模板中的代码会自动添加代码编辑器中。

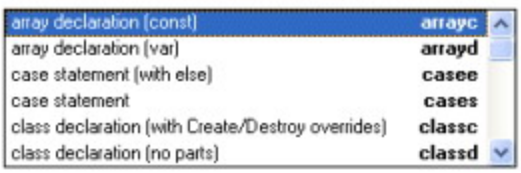


图 15.11　代码列表框

下面笔者详细介绍如何添加代码模板及如何应用代码模板。

（1）在代码模板中单击【Add】按钮可以“添加代码模板”窗口，如图 15.12 所示。

（2）在“添加代码模板”窗口的“Shortcut Name”编辑框中输入一个简码，例如 s，单击【OK】按钮添加模板，如图 15.13 所示。

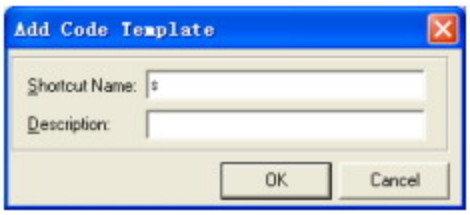


图 15.12　添加代码模板窗口

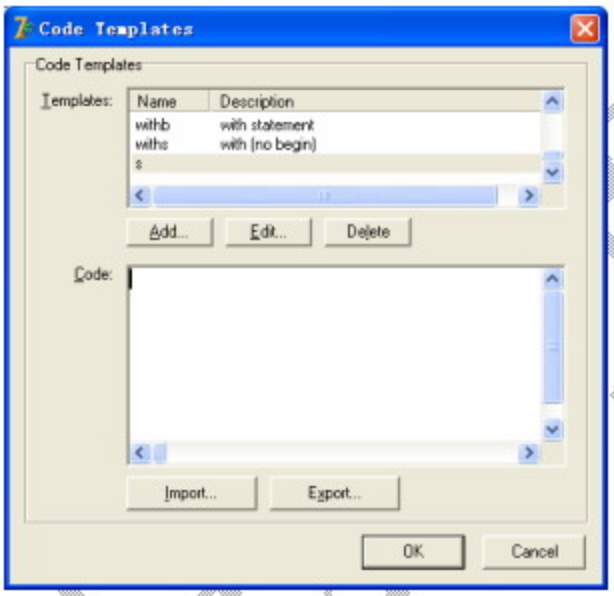


图 15.13　添加模板

（3）在“Code”备注框中添加模板代码，如图 15.14 所示。

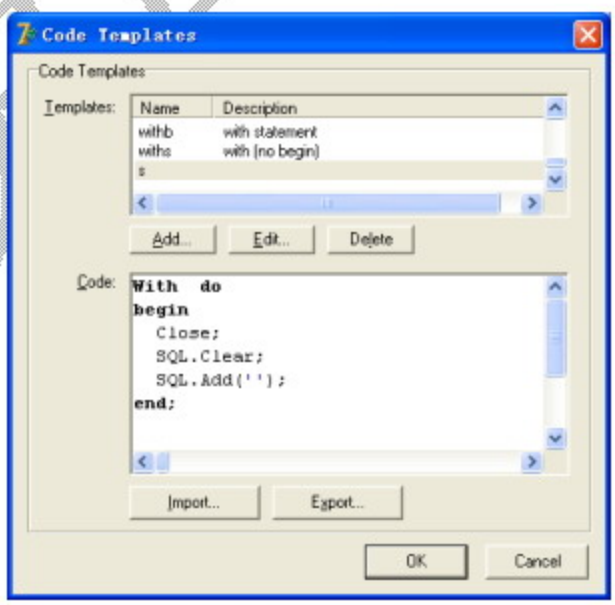


图 15.14　添加模板代码

（4）单击【OK】按钮完成设置。在代码编辑器中按（Ctrl+Shift+J）组合键打开代码列表框，按简码键（本例为s），然后按（Enter）键，模板代码会自动添加到代码编辑器中，如图 15.15 所示。

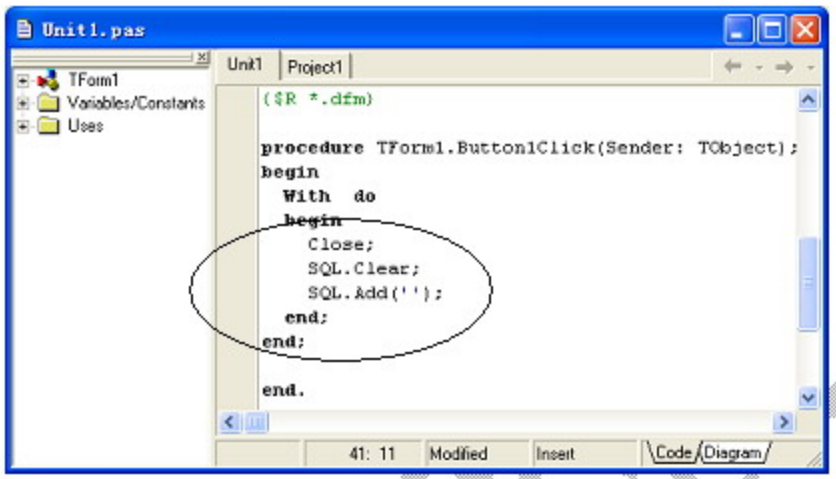


图 15.15　代码编辑器

15.3　代码编辑器显示状态设置

单击系统菜单“Tools”/“Editor Options”选项，将弹出编辑器属性对话框，打开“Display”选项卡，如图 15.16 所示。

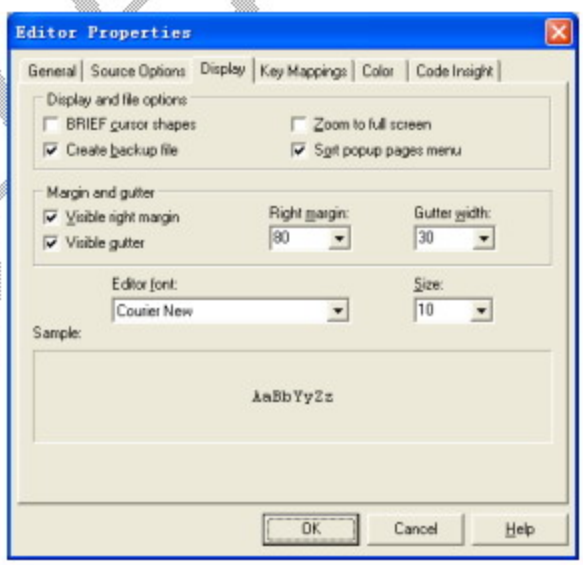


图 15.16　Display 选项卡

在“Display”选项卡中包含了编辑器的显示状态和字体设置，各选项功能如下。

- BRIEF cursor shapes

选中该选项，代码编辑器中的光标会呈一个黑色的方块，如图 15.17 所示。

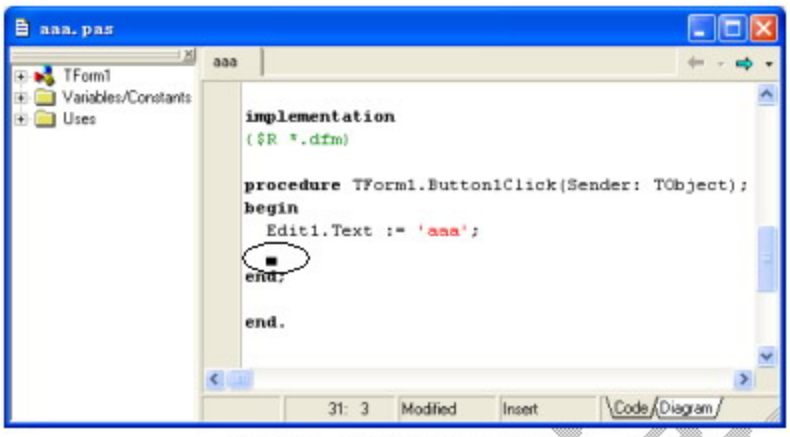


图 15.17　代码编辑器效果图 7

❑ Create backup file

选中该选项，在保存文件时会创建备份文件，备份文件的第一个扩展名为“~”，例如“aaa.~pas”。

❑ Zoom to full screen

选中该选项，代码编辑器会放大到全屏显示，如果该选项没有选中，当代码编辑器最大化时，不能覆盖整个 Windows 窗口。

❑ Sort popup pages menu

选中该选项，在代码编辑器中的单元选项卡中右击，选择“Pages”菜单项，弹出的菜单列表会按字母顺序或单元创建顺序排列。如果该选项没有选中，菜单列表中的选项会按代码编辑器中当前单元选项卡的顺序排列。

❑ Visible right margin

选中该选项，会在代码编辑器右边显示一条线。

❑ Visible gutter

选中该选项，会在代码编辑器左边显示一个边框。

❑ Right margin

该选项用来设置代码编辑器右边线的位置，默认值为 80，输入的数值范围为 0~1024，不在此范围内时会收到一个错误。

❑ Gutter width

该选项用来设置左边框的宽度，默认值为 30。

❑ Editor font

该选项用来设置代码编辑器的字体种类。

❑ Size

该选项用来设置字体的大小。

15.4　语法着色

单击系统菜单“Tools”/“Editor Options”选项，将弹出编辑器属性对话框，打开“Color”选项卡，如图 15.18 所示。

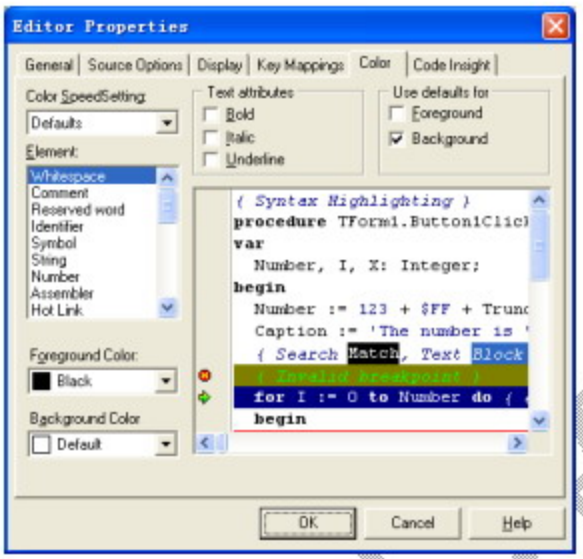


图 15.18　Color选项卡

在“Color”选项卡中包含了程序代码不同成分的颜色设置。各选项功能如下。

❑　Color SpeedSetting

使用该选项可以快速地设置代码编辑器的颜色。在“Color SpeedSetting”下拉列表框中共有 5 个选项，其意义如下。

Defaults：关键字加粗，背景色呈白色。

Classic：关键字呈白色，代码呈黄色，背景呈深蓝色。

Twilight：关键字呈淡蓝色，代码呈白色，背景呈黑色。

Ocean：关键字呈黑色，代码呈深蓝色，背景呈淡蓝色。

Visual Studio：关键字呈蓝色，代码呈黑色，注释呈绿色。

❑　Element

该选项用来对代码中的各个成分进行特殊显示。用户可以在“Element”下拉列表框中选择要设置的代码成分，也可以单击代码编辑器模板中的代码进行选择。

❑　Foreground Color

该选项用来设置被选中的代码成分的前景色。

❑　Background Color

该选项用来设置被选中的代码成分的背景色。

❑　Text attributes

该选项用来设置字体风格。各子选项意义如下。

Bold：字体加粗。

Italic：斜体。

Underline：添加下划线。

❑　Use defaults for

该选项确定是否使用默认的系统颜色设置当前代码成分的前景色和背景色，其子选项意义如下。

Foreground：使用默认的系统颜色设置当前代码成分的前景色。

Background：使用默认的系统颜色设置当前代码成分的背景色。

15.5 代码智能选项设置

单击系统菜单“Tools”/“Editor Options”选项，将弹出编辑器属性对话框，打开“Code Insight”选项卡，如图 15.19 所示。

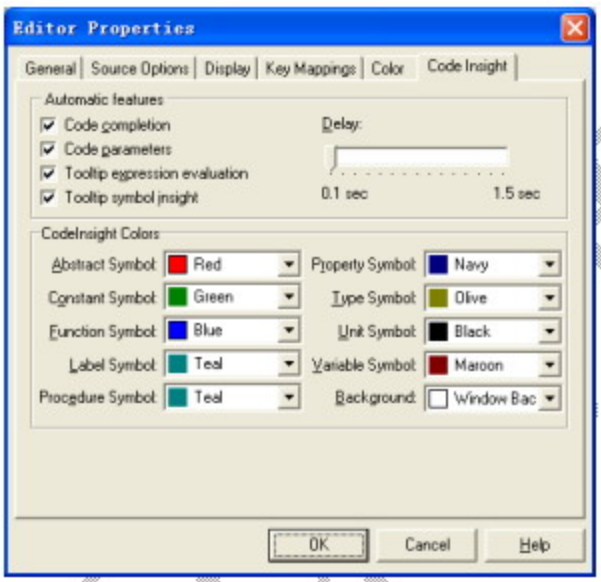


图 15.19 Code Insight 选项卡

在“Code Insight”选项卡中包含了编程过程中经常使用的属性列表设置，各选项功能如下。

❑ Code completion

选中该选项，在类名或对象名后输入一个圆点会显示一个属性列表，如图 15.20 所示。

如果该选项没有选中，可以使用 Ctrl+Space 组合键打开属性列表。



图 15.20 属性列表

❑ Code parameters

选中该选项，在代码编辑器中调用一个方法时，会显示方法的参数列表。如果该选项没有选中，可以使用 Ctrl+Shift+Space 组合键显示参数列表。

❑ Tooltip expression evaluation

选中该选项，在程序调试时，当光标处在某个变量处时会显示变量的值。

❑ Tooltip symbol insight

选中该选项，当光标停留在某个标识处时会显示标识符的声明信息。

❑ Delay

该选项用来设置打开属性列表的时间。

❑ Code Insight Colors

该选项用来设置属性列表中各语法成分（属性、函数、过程等）的颜色。

15.6 环境选项参数设置

单击系统菜单“Tools”/“Environment Options”选项，将弹出环境选项设置对话框，打开“Preferences”选项卡，如图 15.21 所示。

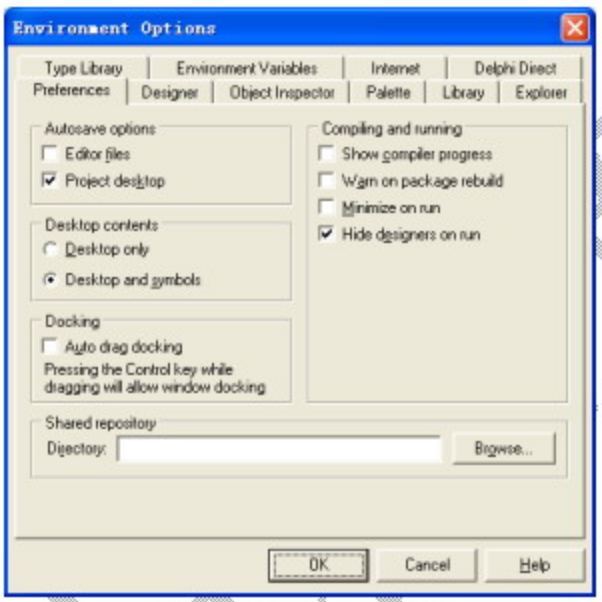


图 15.21 Preferences 选项卡

在“Preferences”选项卡中包含了开发环境的相关设置，各选项功能如下。

- Editor files

选中该选项，当编译或执行应用程序时，程序会保存代码编辑器中的所有文件。

- Project desktop

选中该选项，当退出或关闭一个工程时，程序会保存当前的桌面环境，当下次再打开该工程时，程序会根据上一次的桌面环境设置显示工程。上一次所有被打开的单元文件都将显示出来，不论单元文件是否在工程中使用。例如，当前工程只有一个单元文件（aaa.pas），在当前工程中打开另一个工程的单元文件（bbb.pas），如果“Projcet desktop”选项被选中，关闭当前工程后，再次打开该工程时，“bbb.pas”单元文件也会被显示在代码编辑器中。

- Desktop only

当“Project desktop”选项被选中时，该选项用来保存桌面信息。例如，代码编辑器中的单元文件、当前打开的窗口。

- Desktop and symbols

当“Project desktop”选项被选中时，该选项用来保存桌面信息和上一次成功编译的符号信息。下面是一个工程按钮的单击事件代码。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.Text := '姓名';
```

```
    Edit2.Text :='张明';
end;
```

保存该工程，修改该按钮的单击事件代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Edit1.Text := '姓名';
    Edit2.Text :='王雨';
end;
```

运行程序，单击按钮，效果如图 15.22 所示。

结束应用程序，关闭工程，不保存对工程所做的修改。重新打开该工程，会出现如图 15.23 所示的对话框。

运行程序，单击按钮，运行结果与图 15.22 相同。如果单击系统菜单“Project”/“Build”菜单项，然后重新运行程序，单击按钮，运行结果则如图 15.24 所示。



图 15.22　运行结果



图 15.23　加载编译器符号信息



图 15.24　运行结果

- Auto drag docking

选中该选项，可以将两个工具对话框合并在一起。例如，将树视图对话框拖到对象观察器对话框上，会发现两个对话框合并成了一个对话框，如图 15.25 所示。

- Show compiler progress

选中该选项，在编译应用程序过程中会出现编译报告，如图 15.26 所示。

- Warn on package rebuild

选中该选项，在重新编译包时，会出现警告信息，如图 15.27 所示。

- Minimize on run

选中该选项，运行应用程序时会最大化工程，当应用程序结束时恢复工程。

- Hide designers on run

选中该选项，运行应用程序时会隐藏设计窗口。例如，选中该选项，运行程序时，会发现对象观察器和窗体消失了，当应用程序结束时，对象观察器和窗体又显示出来。

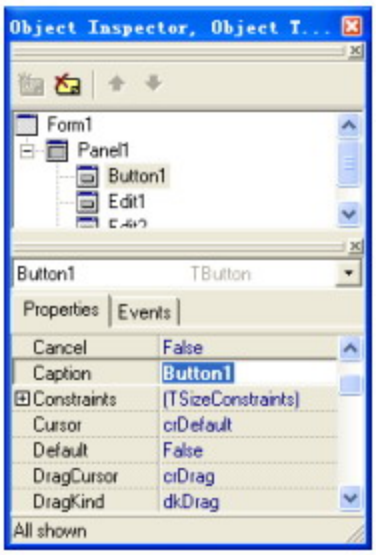


图 15.25　合并对话框

- Directory

该选项用来设置工程查找智能库文件（delphi32.dro）的路径，单击 Browse 按钮可以设置自己的查找路径，智能库文件通常存放在 Bin 目录下。



图 15.26　编译报告



图 15.27　警告信息

15.7　窗体和组件设置

单击系统菜单“Tools”/“Environment Options”选项，将弹出环境选项设置对话框，打开“Designer”选项卡，如图 15.28 所示。

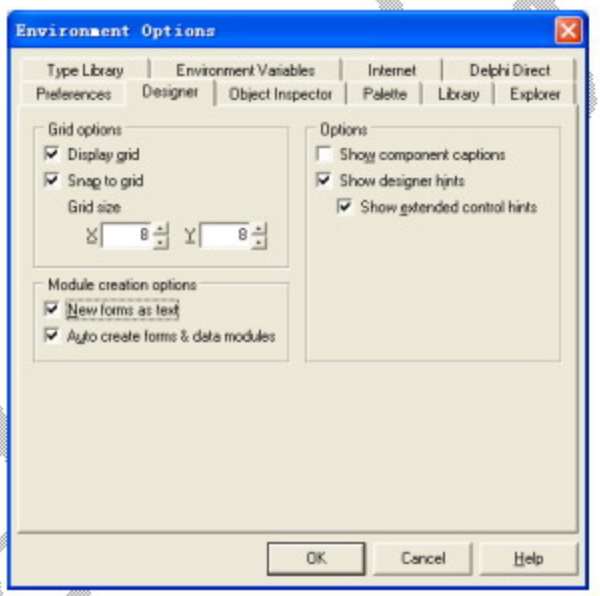


图 15.28　Designer 选项卡

在“Designer”选项卡中包含了窗体和组件的相关设置，各选项意义如下。

- Display grid

选中该选项，在窗体上会显示圆点。

- Snap to grid

选中该选项，当向一个窗体放置组件时，组件会自动对齐到最近的网格线，而不会在两个网格线之间。

- Grid size

该选项以像素为单位设置网格大小。数值范围为 2~128。

- New forms as text

该选项确定窗体文件的保存格式。工程中的窗体文件只能以两种文件格式保存，分别为二进制文件和文本文件。文本文件很容易被其他工具修改，而且容易使用版本控制

系统进行管理。二进制文件是为了和早期的版本兼容而保留的。选中该选项，新建的窗体文件可以使用记事本打开，可以在窗体上单击鼠标右键，在弹出的快捷菜单中将“Text DFM”或“Text XFM”命令置为选中或非选中状态可将窗体文件在文本文件和二进制文件之间切换。

- Auto create forms & data modules

该选项用来确定程序运行时是否自动创建窗体，如果该选项没有选中，新建的窗体在程序运行时不会构建实例，需要使用代码构建实例，在工程选项对话框的“Forms”选项卡中，会发现新建的窗体在“Available forms”列表框中。

- Show component captions

选中该选项，将不可视组件放置在窗体上时，会显示不可视组件的标题。

- Show designer hints

选中该选项，将鼠标停留在组件上时，会显示提示信息，即显示组件的类名。该选项并不影响组件面板上的提示信息。

- Show extended control hints

选中该选项，当鼠标停留在控件上时，会显示控件的附加信息，例如，控件的位置、控件大小、添加到窗体上的控件顺序等信息。如果“Show designer hints”选项没有选中，该选项将不可用。

15.8 个性化开发环境设置

根据前面几节对 Delphi 代码编辑器和开发环境的介绍，同时，为了给程序员提供一个方便、快捷的开发环境，现将 Delphi 开发环境设置如下。

- 代码编辑器设置

单击系统菜单“Tools”/“Editor Options”选项，将弹出编辑器属性对话框，打开“Color”选项卡，如图 15.29 所示。

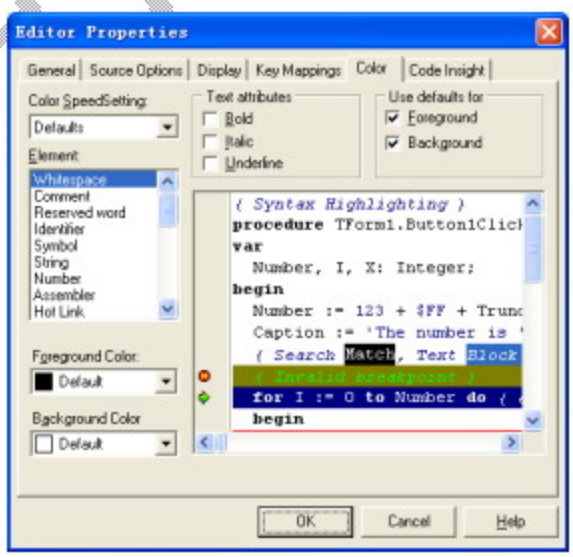


图 15.29　Color选项卡

在“Element”下拉列表框中选中“Comment”选项，在“Foreground Color”下拉列表框中设置颜色为“Fuchsia”；在“Element”下拉列表框中选中“String”选项，在“Foreground Color”下拉列表框中设置颜色为“Red”；在“Element”下拉列表框中选中“Number”选项，在“Foreground Color”下拉列表框中设置颜色为“Blue”，单击 OK 按钮完成设置，代码编辑器效果如图 15.30 所示。

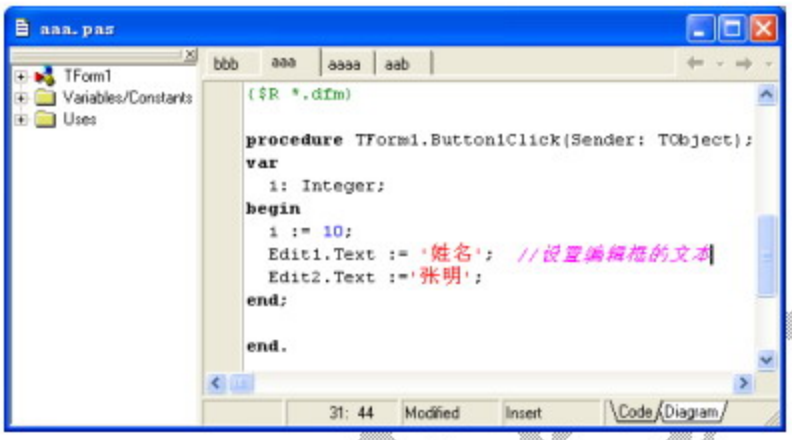


图 15.30　代码编辑器效果图 8

❑　窗体设置

单击系统菜单“Tools”/“Environment Options”选项，将弹出环境选项设置对话框，打开“Designer”选项卡，在该选项卡中设置网格的横、纵坐标为 2，缩小网格，以便在利用鼠标拖动控件时，能够将控件精确地拖动到某个位置。选项设置如图 15.31 所示。

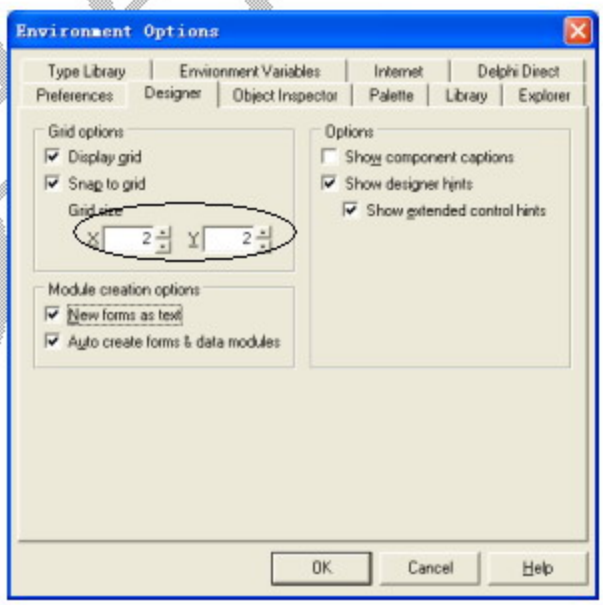


图 15.31　环境选项对话框 3

单击【OK】按钮完成设置。图 15.32、图 15.33 显示了设置前和设置后窗体的不同效果。



图 15.32　网格缩小前的窗体效果

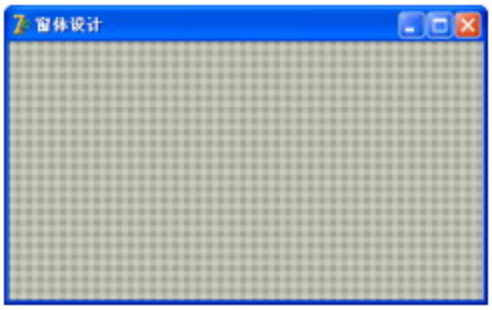


图 15.33　网格缩小后的窗体效果

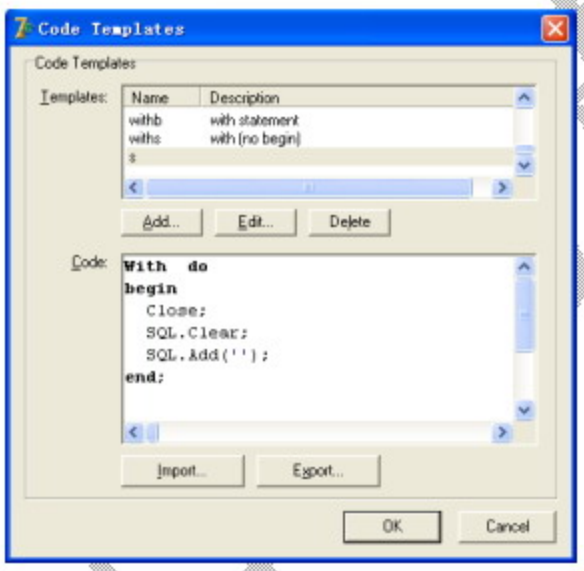


图 15.34　添加模板代码

单击【OK】按钮完成设置。在代码编辑器中按〈Ctrl+Shift+J〉组合键打开代码列表框，按简码键（本例为 s），然后按〈Enter〉键，模板代码会自动添加到代码编辑器中，如图 15.35 所示。

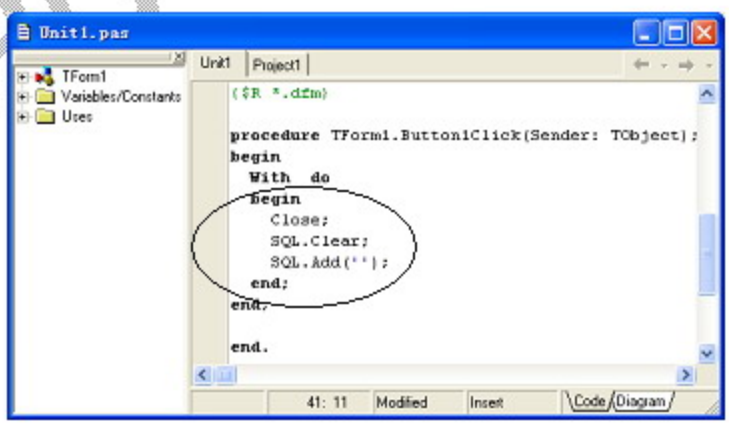


图 15.35　代码编辑器

第16章 Delphi 7在物流配送管理系统中的应用

16.1 物流配送管理系统概述

16.1.1 开发背景

随着我国国民经济的不断发展，企业规模不断扩大，企业间的竞争也越来越激烈。为了在激烈的竞争中赢得发展，有些企业在提高产品质量、采取优惠措施的同时，还提供了送货上门服务，为用户带来许多方便。但是，也给企业带来一些困难，因为企业需要额外投入大量人力、财力进行这些服务，而且企业管理也会变得愈来愈复杂。计算机管理在很大程度上可以帮助企业克服这些困难，它可以实现商品销售、配送一条龙管理，节省大量人力、财力资源，使企业管理变得科学有序，物流经营清晰、透明。

16.1.2 运行环境

本软件适用于 Microsoft Windows 98/XP/2000 等操作系统，CPU 要求 Intel Pentium 166MHz 或更高的处理器，内存 64MB 以上。

16.2 系统调查

16.2.1 手工流程

在传统的企业管理模式中，商品配送流程比较繁琐，一般需要经过 5 个步骤，如图 16.1 所示。

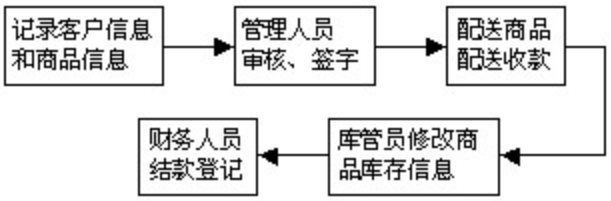


图 16.1　手工流程图

传统的企业管理主要以人工管理为主，由于以手工填写、计算数据，在各个环节中很容易出现错误，并且如果一个环节出现问题，整个管理过程就会混乱。

16.2.2　需求分析

万通集团是一个以营销食品为主的企业，该企业为了提高竞争力，向客户提供了送货上门服务。由于在送货过程中需要进行送货申请、送货审核、送货结款等流程，人工管理相当繁琐，而且容易出错。2003 年 8 月，该企业决定采用计算机对商品送货进行管理。

16.3　系 统 分 析

16.3.1　系统功能

- 商品信息管理

在商品信息管理中需要实现商品信息的添加、修改、删除。

- 客户信息管理

在客户信息管理中需要实现客户信息的添加、修改、删除。

- 库存管理

在库存管理中需要实现商品入库。

- 操作员管理

在操作员管理中需要实现操作员信息的添加、删除，并且能够设置操作员权限。

- 配送申请管理
- 配送审核管理
- 配送开票管理
- 配送查询管理

系统结构如图 16.2 所示。

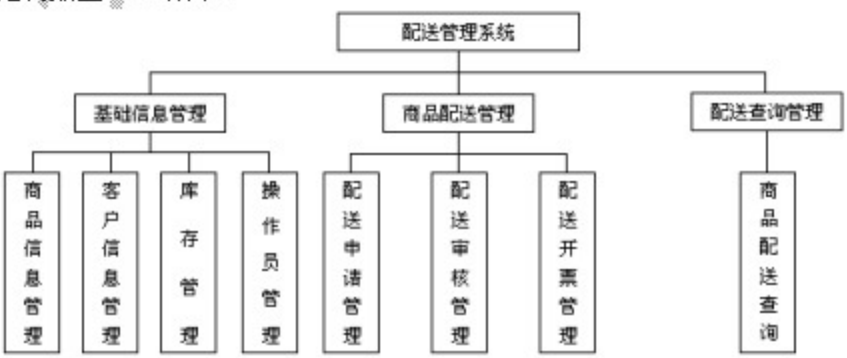


图 16.2　配送管理系统结构图

16.3.2 业务流程

具体的业务流程如图 16.3 所示。



图 16.3 配送管理系统业务流程图

16.4 数据库设计

16.4.1 数据库概念设计

配送管理系统的 E-R 图如图 16.4 所示。



图 16.4 配送管理系统 E-R 图

16.4.2 数据库逻辑结构设计

配送管理系统共使用了 8 张数据表，分别为操作员信息表、配送票号表、客户基础信息表、配送申请表、库存表、商品基础信息表、客户结款表、客户结款记录表。由于篇幅关系，在此只列出使用较多的数据表结构（表 16.1 和表 16.2），其他表结构请参考光盘。

表 16.1 配送票号表

字段名称	字段类型	主键	外键	是否为空	默认值
配送票号	Varchar(30)	是		否	
品种数	Int				
数量	Int				
应付金额	Money				
配送时间	DateTime				GetDate()
操作员	Varchar(20)				
审核负责人	Varchar(20)				
完成送货	Int				
是否可用	Int				
开票人	Varchar(30)				

表 16.2　　配送申请表

字 段 名 称	字 段 类 型	主　键	外　键	是否为空	默 认 值
客户编号	Varchar(30)				
客户全称	Varchar(30)				
商品编号	Varchar(30)				
商品名称	Varchar(30)				
数量	Int				
金额	Money				
折扣	Real				
配送票号	Varchar(30)		是		

本系统使用 Microsoft 的 SQL Server 2000 作为后端数据库。因此，有必要对该数据库做简要介绍。SQL Server 2000 是 Microsoft 开发的功能强大的关系型数据库，它是专为 Windows 操作系统设计的数据库管理系统，在开发 Windows 应用程序时，SQL Server 2000 会比其他跨平台的数据库管理系统功能更强大，操作更简单。下面将以创建操作员信息表为例介绍如何使用 SQL Server 2000 创建数据表。

（1）打开企业管理器，展开各菜单项，用鼠标右键单击“数据库”选项，将弹出一个快捷菜单，如图 16.5 所示。

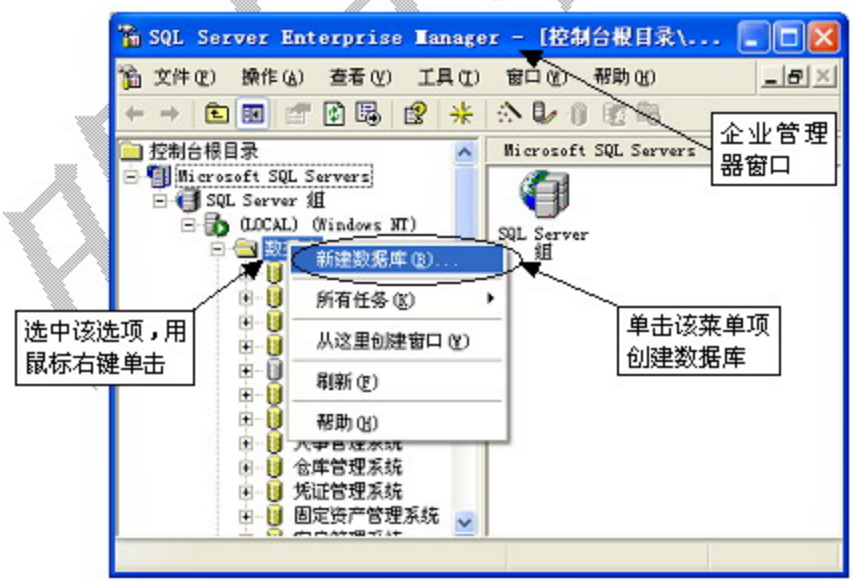


图 16.5　创建数据库

（2）单击“新建数据库”菜单项，弹出如图 16.6 所示的对话框。

（3）在“名称”编辑框中输入所要创建的数据库名称，例如“客户管理系统”。单击“确定”按钮创建数据库。

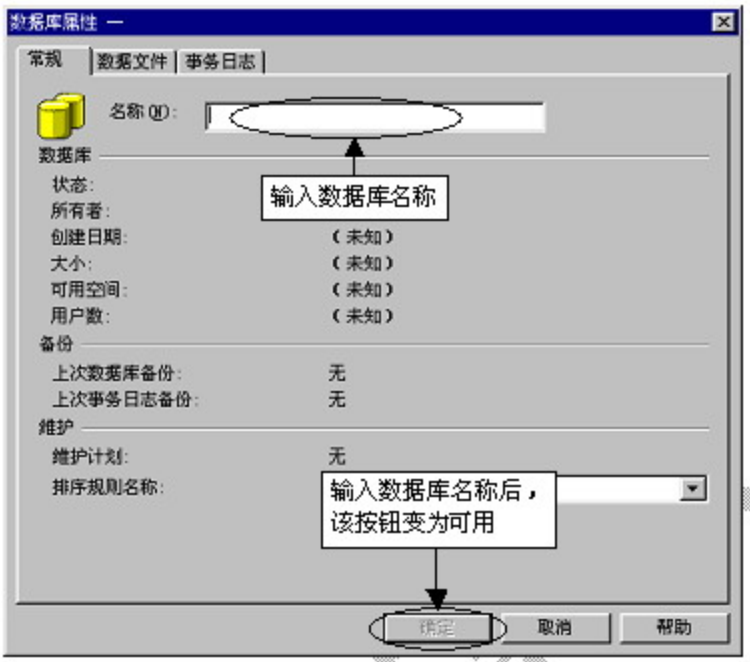


图 16.6　数据库属性

（4）展开新建的“客户管理系统”数据库，用鼠标右键单击“表”菜单项，将弹出如图 16.7 所示的窗口。

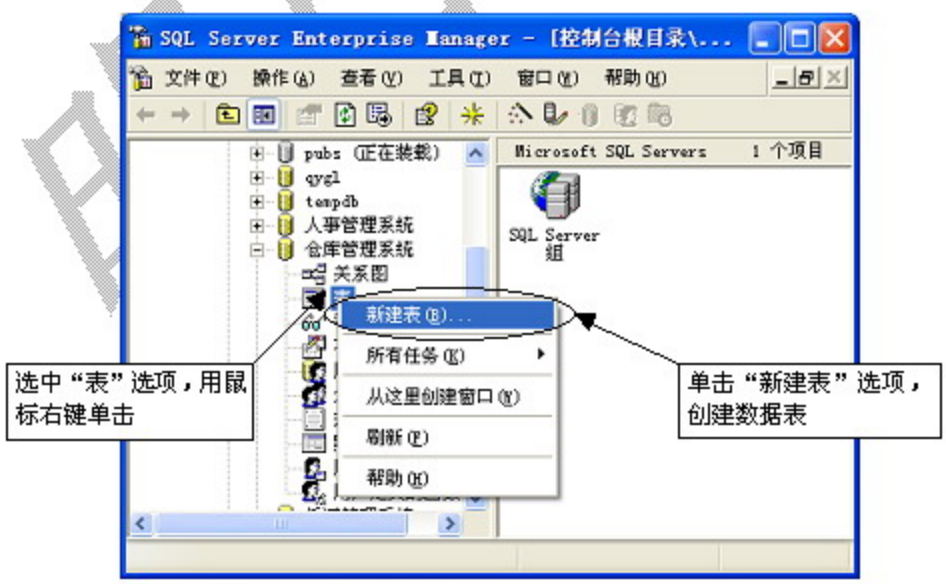


图 16.7　创建数据表

（5）单击“新建表”菜单项，弹出如图 16.8 所示的窗口。

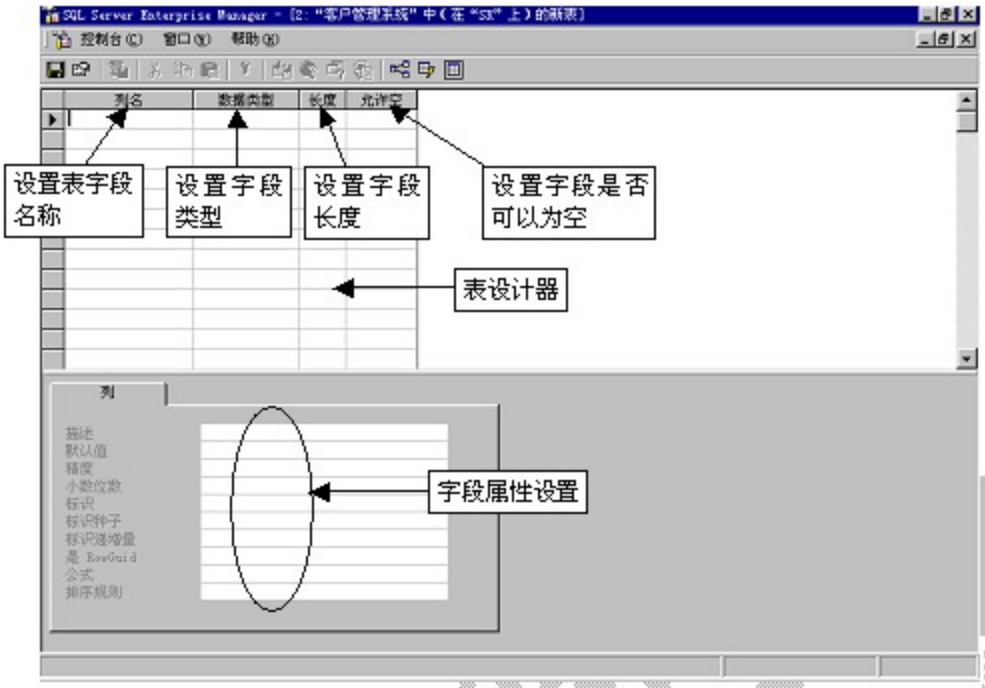


图 16.8　创建数据表

（6）在“列名”字段中输入字段名称，分别为操作员编号、操作员姓名、操作员密码、操作员级别、时间。在“数据类型”字段选择字段的数据类型，分别为 varchar、varchar、varchar、varchar、datetime。在“长度”字段输入字段长度，分别为 20、20、20、20、8。效果如图 16.9 所示。

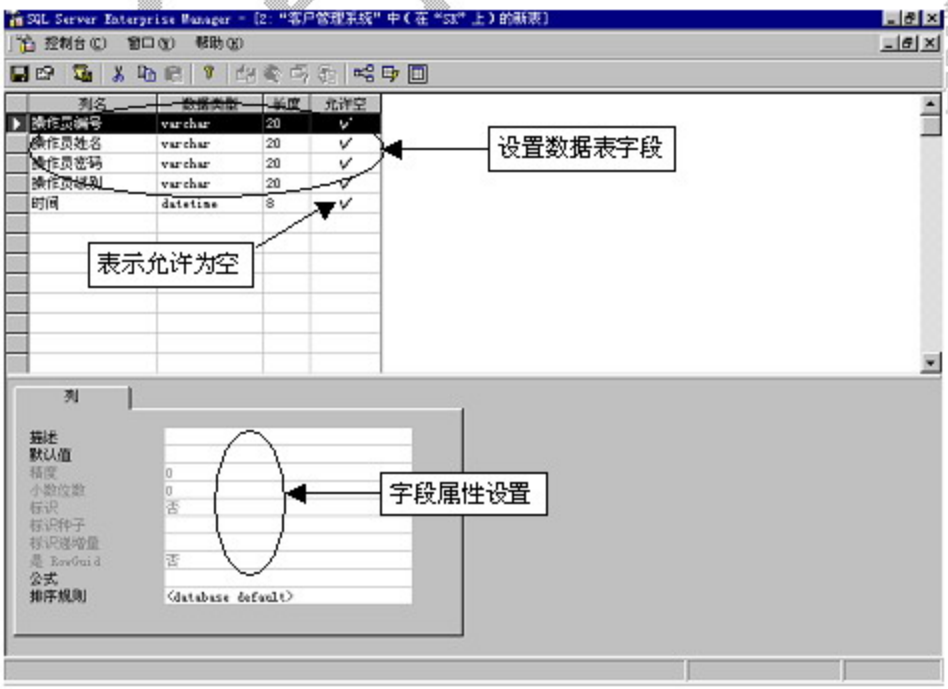


图 16.9　添加字段

（7）选中“操作员编号”字段，用鼠标右键单击，弹出一个快捷菜单，效果如图16.10所示。

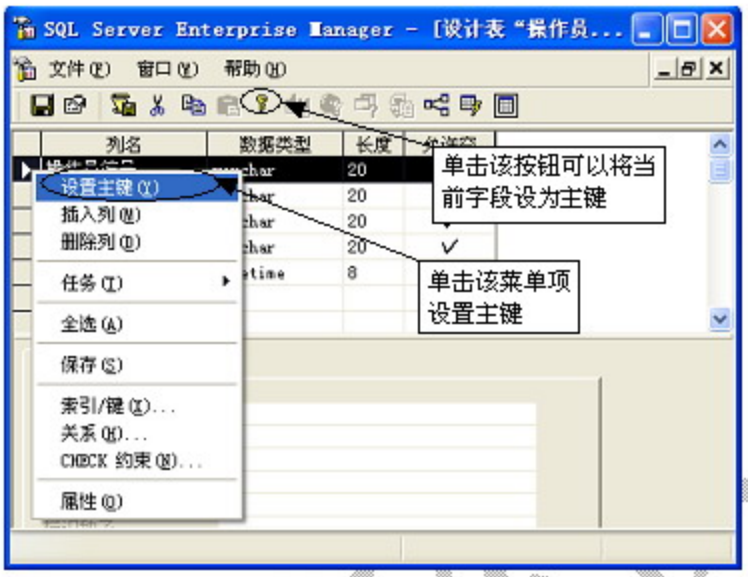


图16.10 设置主键

（8）单击“设置主键”菜单项，将“操作员编号”字段设为主键。选中“时间”字段，在“默认值”选项中输入字段默认值“GetDate()”，即服务器当前时间。

（9）单击工具栏上的按钮，将数据表保存为“操作员信息表”。至此完成操作员信息表的创建。

16.5 处理过程分析

16.5.1 配送审核处理过程分析

配送审核是对配送信息进行审核，所以在审核前应先调出配送信息。需要注意的是，查询的配送信息应该是没有审核或审核没有通过的配送信息，不能是已完成审核或已完成送货的审核信息。在审核时需要记录审核人、审核是否通过等信息。配送审核业务流程如图16.11所示。

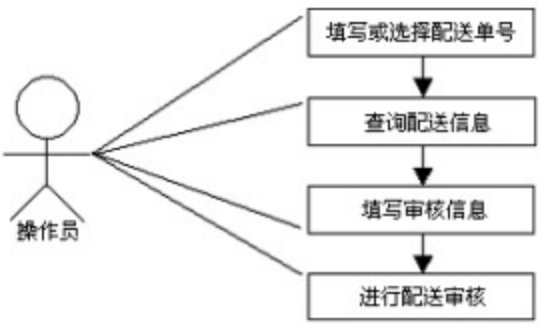


图16.11 配送审核开发业务流程

16.5.2 配送开票处理过程分析

配送开票管理是对配送出去的商品进行结款。与配送审核一样，配送开票首先需要将配送信息查询出来，然后录入结款信息，例如：开票人、结款金额等，最后修改商品库存，保存结款信息，将该条配送信息标记为已结款。配送开票业务流程如图16.12所示。

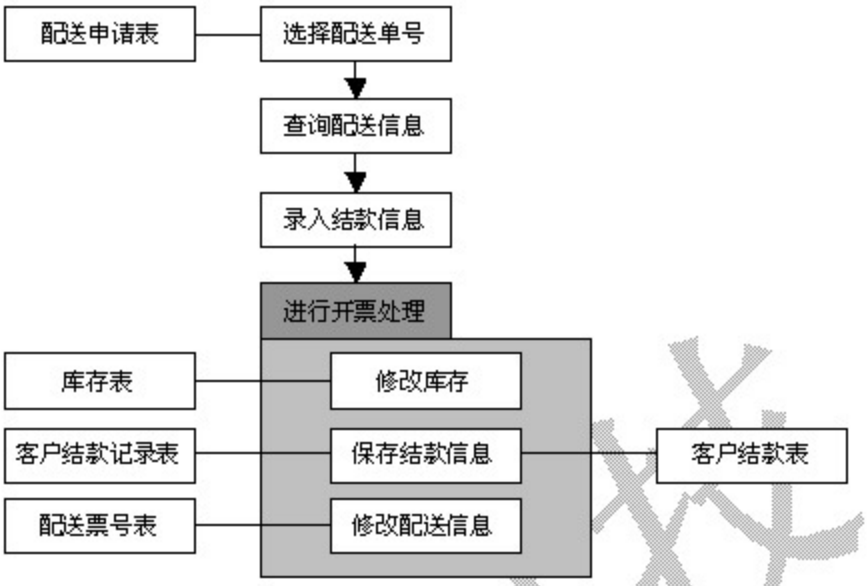


图 16.12　配送开票业务流程

16.6　主程序设计

16.6.1　实现目标

- 在程序启动时进行身份验证，只有操作员名称和密码都正确才能进入系统。
- 通过系统菜单或工具栏按钮可以调用响应子程序。
- 在程序中以列表形式显示程序模块，双击列表中的程序模块，可以将程序模块中的各个子程序以图标形式显示出来，双击子程序图标可以打开相应子程序。
- 在状态栏中显示当前登录的用户名称和登录时间。

程序界面如图 16.13 所示。



图 16.13　主窗体界面

16.6.2 建立主窗体

主窗体设计步骤如下。

（1）打开 Delphi 7，在默认情况下，Delphi 会创建一个含有窗体的工程，在该窗体上放置 TMainMenu、TImageList、TListView、TCoolBar、TToolBar、TSpeedButton、TStatusBar 等组件，设置各主要组件属性，如表 16.3 所示。

表 16.3　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form90	TForm	Font.Color	clBlue
		Font.Size	8
		ShowLines	True
SpeedButton1	TSpeedButton	Caption	操作员
ToolBar1	TToolBar	Images	ImageList2

（2）在窗体上双击 TImageList 组件，将弹出图像列表编辑框，单击 Add 按钮，可以将位图或图标加到 TImageList 中，如图 16.14 所示。



图 16.14　图像列表编辑框

（3）在窗体上双击 TMainMenu 组件，将弹出菜单设计窗口。在弹出菜单的底部和右侧各有一个虚边框，选中该虚边框，通过修改对象观察器中的“Caption”属性，即可完成系统菜单设置。效果如图 16.15 所示。

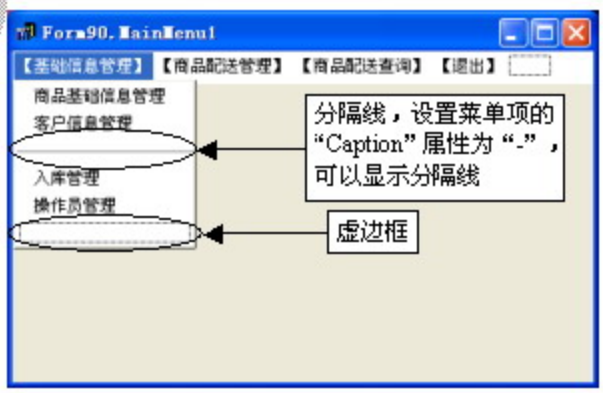


图 16.15　系统菜单

本系统中使用的菜单部分属性设置如表 16.4 所示。

表 16.4　　菜单属性设置

对 象 名	对 象 类 型	属　性	值
MainMenu1	TMainMenu	Images	ImageList1
N3	TMenuItem	Caption	基础信息管理
N4	TMenuItem	Caption	商品基础信息管理
		ImageIndex	3
N5	TMenuItem	Caption	客户信息管理
		ImageIndex	4
N10	TMenuItem	Caption	-
N8	TMenuItem	Caption	入库管理
		ImageIndex	5
N9	TMenuItem	Caption	操作员管理
		ImageIndex	1
N1	TMenuItem	Caption	商品配送管理
N13	TMenuItem	Caption	商品配送申请
		ImageIndex	0
N11	TMenuItem	Caption	-
N14	TMenuItem	Caption	商品配送审核
		ImageIndex	1

（4）在窗体上双击 TStatusBar 组件，打开状态栏面板编辑器，单击 Add New 按钮添加面板。选中面板选项，通过修改对象观察器中的“Text”属性设置面板标题。效果如图 16.16 所示。

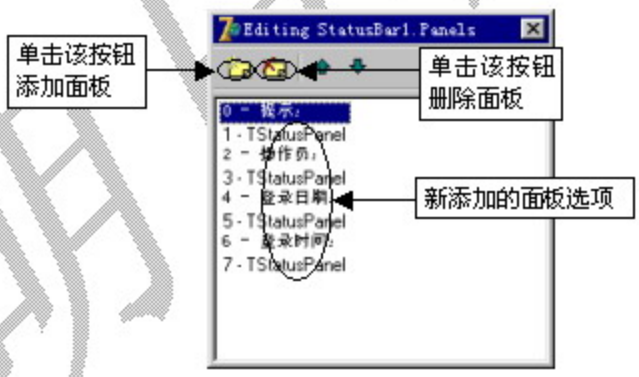


图 16.16　面板编辑器

16.6.3　程序相关代码

下面是菜单按钮的单击事件，单击菜单中的按钮会调用相应的子程序，由于篇幅关系，在此只列出部分菜单按钮的单击事件。

```
procedure TForm90.N13Click(Sender: TObject);
begin
   Application.CreateForm(TForm29,Form29);
   Form29.ShowModal;
   Form29.Free;
end;
```

明日科技

```
procedure TForm90.N14Click(Sender: TObject);
begin
  Application.CreateForm(TForm31,Form31);
  Form31.ShowModal;
  Form31.Free;
end;
procedure TForm90.N2Click(Sender: TObject);
begin
  Application.CreateForm(TForm32,Form32);
  Form32.ShowModal;
  Form32.Free;
end;
```

在主窗体显示时设置状态栏信息，调用登录窗口。

```
procedure TForm90.FormShow(Sender: TObject);
begin
  StatusBar1.Panels[5].Text := FormatDateTime('yyyy-mm-dd',Now);//显示系统日期
  StatusBar1.Panels[7].Text := FormatDateTime('hh:mm:ss',Now);//显示系统时间
  Application.CreateForm(TForma,Forma); //创建系统登录窗口
  Forma.ShowModal;
  Forma.Free;
end;
```

16.7 数据模块设计

数据模块类似于普通的窗体，在数据模块上只能存放不可视组件，例如 TTimer、TOpenDialog 等。我们可以将数据访问组件、BDE 组件或 ADO 组件放在数据模块中，程序中的单元可以访问数据模块中的组件，这样做有利于程序的结构化，便于修改。下面介绍如何建立数据模块。

（1）启动 Dephi 7，打开工程“psgl”，单击“File”/“New”/“Data Module”菜单项，打开创建数据模块。

（2）选中数据模块，在对象观察器中修改“Name”属性为“DataModule1”。

（3）在数据模块上放置 TADOConnection 组件、TADOQuery 组件，如图 16.17 所示。



图 16.17 数据模块

（4）设置各主要组件属性如表 16.5 所示。

表 16.5 组件属性设置

对 象 名	对 象 类 型	属 性	值
ADOConnection1	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=配送管理系统
ADOQuery1	TADOQuery	Connection	ADOConnection1
ADOQuery2	TADOQuery	Connection	ADOConnection1
ADOQuery3	TADOQuery	Connection	ADOConnection1

16.8 程序设计与编码

16.8.1 配送申请管理设计

1. 概述

配送申请是进行商品配送的第一步，当客户提出配送商品时，由操作员进行登记，产生配送票号，记录客户信息及商品配送信息，将其保存到数据表中。配送申请运行结果如图 16.18 所示。

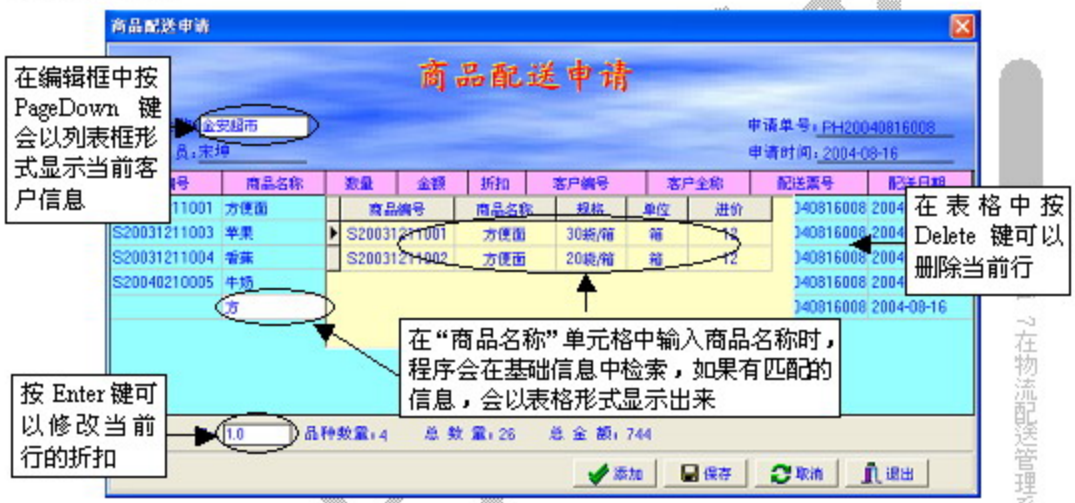


图 16.18 配送申请运行结果

2. 设计步骤

（1）创建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TStringGrid、TDataSource、TDBGrid、TListBox、TShape 等组件，如图 16.19 所示。

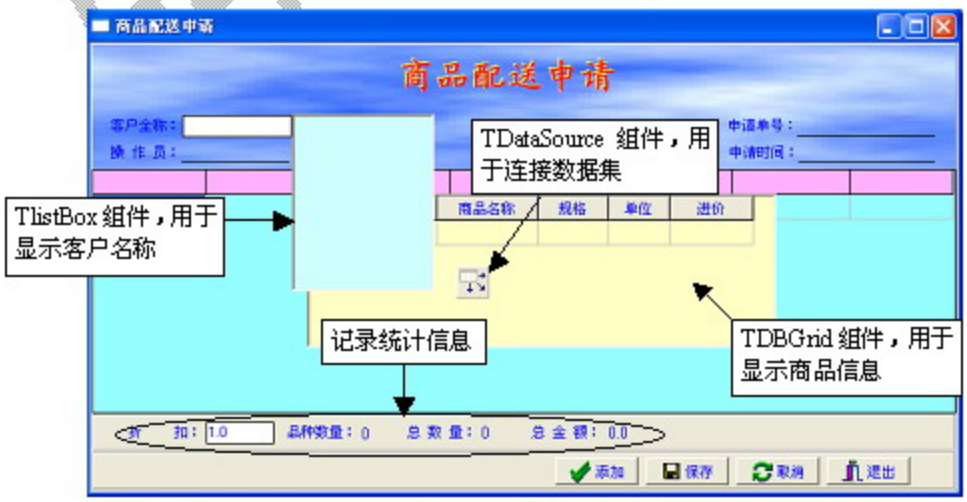


图 16.19 商品销售管理

（2）设置各主要组件属性如表 16.6 所示。

表 16.6　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form29	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
ListBox1	TListBox	Visible	False
DBGrid1	TDBGrid	Visible	False
		DataSource	DataSource1
Edit8	TEdit	Text	1.0

3. 程序相关代码

在窗体创建时设置 StringGrid1 列标题。

```
procedure TForm29.FormCreate(Sender: TObject);
begin
  StringGrid1.Cells[0,0]:='        商品编号';
  StringGrid1.Cells[1,0]:='        商品名称';
  StringGrid1.Cells[2,0]:='       数量';
  StringGrid1.Cells[3,0]:='       金额';
  StringGrid1.Cells[4,0]:='     折扣';
  StringGrid1.Cells[5,0]:='          客户编号';
  StringGrid1.Cells[6,0]:='       客户全称';
  StringGrid1.Cells[7,0]:='          配送票号';
  StringGrid1.Cells[8,0]:='       配送日期';
end;
```

在窗体显示时进行数据初始化，包括清理界面上控件的文本（数据），向列表框中加载客户信息（当“客户全称”编辑框获得焦点时，按 PageDown 键会以列表框显示客户信息）。

```
procedure TForm29.FormShow(Sender: TObject);
begin
  Clear; //调用Clear过程进行界面数据初始化
  Label9.Caption := czymc;
  ListBox1.Clear; //清空列表框中的数据
  //向ListBox1中添加客户全称
  begin
    with DataModule1.ADOQuery1 do
    begin
      Close;//关闭数据集
      SQL.Clear;//清空SQL语句
      SQL.Add('select distinct 客户全称 from 客户基础信息表');//添加SQL语句
      Open;//打开数据集
    end;
    //利用循环，向列表框中添加客户全称
    while not DataModule1.ADOQuery1.Eof do
```

```
      begin
        ListBox1.Items.Add(DataModule1.ADOQuery1.FieldByName('客户全称').AsString);
        DataModule1.ADOQuery1.Next;
      end;
    end;
    BitBtn3.SetFocus;
  end;
```

以下是“保存”按钮的单击事件，用户单击“保存”按钮会将列表中的数据保存到数据库中。在配送申请时，主要向“配送票号表”和“配送申请表”中保存数据。

在保存过程中，首先判断列表数据是否有误，然后使用事务，保证在保存过程中数据的完整性。

```
procedure TForm29.BitBtn4Click(Sender: TObject);
var
  i: Integer;
begin
  if Jcxmlb = False then //检查列表是否有误即判断表格中是否有空的数据项
  begin
    Application.MessageBox('项目列表有误。','提示',0+64);
    Exit;
  end;
DataModule1.ADOConnection1.BeginTrans; //开始一个事务
  try
  with DataModule1.ADOQuery1 do
  begin
    Close;//关闭数据集
    SQL.Clear;//清空SQL语句
    //添加SQL语句
    SQL.Add('Insert 配送票号表 values
  (:a,:aa,:b,:c,Default,:d,"",Default,Default,"")');
  //为参数赋值
    Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[7,1]);
    Parameters.ParamByName('aa').Value := StrToInt(Label16.Caption);
    Parameters.ParamByName('b').Value := StrToInt(Label17.Caption);
    Parameters.ParamByName('c').Value := StrToFloat(Label18.Caption);
    Parameters.ParamByName('d').Value := Trim(Label9.Caption);
    ExecSQL;//执行SQL语句
  end;
  //利用循环将表格中的信息保存到配送申请表中
  For i := 1 to StringGrid1.RowCount-1 do
  begin
    with DataModule1.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Insert 配送申请表 Values(:a,:b,:c,:d,:e,:f,:g,:h)');
      Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,i]);
```

```
            Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[1,i]);
            Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[5,i]);
            Parameters.ParamByName('d').Value := Trim(StringGrid1.Cells[6,i]);
            Parameters.ParamByName('e').Value := StrToInt(StringGrid1.Cells[2,i]);
            Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[3,i]);
            Parameters.ParamByName('g').Value := StrToFloat(StringGrid1.Cells[4,i]);
            Parameters.ParamByName('h').Value := Trim(StringGrid1.Cells[7,1]);
            ExecSQL;
          end;
        end;
        DataModule1.ADOConnection1.CommitTrans; //提交事务
        Application.MessageBox('保存成功。','提示',0+64);
        Clear;
        BitBtn3.SetFocus;
        Except
          DataModule1.ADOConnection1.RollbackTrans;//回滚事务
          Application.MessageBox('系统出错。','提示',0+64);
          Close; //关闭窗体
        end;
      end;
```

为了提高用户录入数据的灵活性，在程序中应允许用户录入多行数据，并且对于不需要的数据，用户可以删除，以下是该功能的实现部分。

在删除数据时，首先判断表格中的数据是否超过两行（首行为标题），如果没有，则清空当前行，如果表格中数据超过两行，则将当前行之后的每一行数据赋给上一行数据，同时清空并删除最后一行。

在添加数据时，如果表格中没有空行，并且当前行处于最后一行，按下“↓”键添加新行。

```
procedure TForm29.StringGrid1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
  R,L: Integer;
  cp: String;//  记录当前行的价格
  cl: String;//  记录当前行的数量
begin
  CloseForm(Key);// CloseForm是一个自定义过程，按Esc键可以关闭窗体
  S1 := False;
  ss := False;
  cp := '';
  cl:= '';
  if key = VK_Delete then //按Delete键可以删除当前行
  begin
    if StringGrid1.RowCount>2 then //如果表格有多行数据,将当前行之后的数据赋给它上一行
    //数据,然后清空并删除最后一行数据,一定要清空最后一行数据再删除,否则,在添加新行时会
    //将最后一行数据显示在上面
    begin
```

```
        If Application.MessageBox('确实要删除该条记录吗？','提示',MB_YESNO )= ID_Yes then
        begin
          if IsNull = False then
          begin
            cl := StringGrid1.Cells[2,x];
            cp := StringGrid1.Cells[3,x];
            if CurrentIsCf = False then
              Label16.Caption := IntToStr(StrToInt(Label16.Caption)-1);
          end;
          ClearString;
          if x <> StringGrid1.RowCount-1 then
          begin
            For r := x+1 to StringGrid1.RowCount-1 do //将当前行之后的数据赋给上一行数据
              For l := 0 to StringGrid1.ColCount-1 do
                StringGrid1.Cells[l,r-1]:= StringGrid1.Cells[l,r];
          end;
          ClearStringEnd;//清空最后一行数据
          StringGrid1.RowCount := StringGrid1.RowCount-1;//使表格行数减一
          if Trim(cl)<>'' then
            Label17.Caption := IntToStr(StrToInt(Label17.Caption)- StrToInt(cl));
          if Trim(cp)<>'' then
            Label18.Caption := FloatToStr(StrToFloat(Label18.Caption)-StrToFloat(cp));
          StringGrid1.SetFocus;
          StringGrid1.Col := 2;
          StringGrid1.Col := 1;
          Exit;
        end;
      end
      else if   StringGrid1.RowCount = 2 then //如果表格只有一行数据,清空该行数据
      begin
        If Application.MessageBox('确实要删除该条记录吗？','提示',MB_YESNO )= ID_Yes then
        begin
          ClearString;
          Label16.Caption := '0';
          Label17.Caption :='0';
          Label18.Caption := '0.0';
          StringGrid1.SetFocus;
          StringGrid1.Col := 2;
          StringGrid1.Col := 1;
        end;
      end;
    end;
    if (key = Vk_Next)and(DBGrid1.Visible = True)then
    begin
      DBGrid1.SetFocus;
      Exit;
    end;
```

```
    //按下方向键添加新行
    if (Key = VK_Down)and(IsNull = False)and(x = StringGrid1.RowCount-1) then
    begin
      StringGrid1.RowCount := StringGrid1.RowCount+1;
      StringGrid1.Cells[5,StringGrid1.RowCount-1]:=
        StringGrid1.Cells[5,StringGrid1.RowCount-2];
      StringGrid1.Cells[6,StringGrid1.RowCount-1]:=
        StringGrid1.Cells[6,StringGrid1.RowCount-2];
      StringGrid1.Cells[4,StringGrid1.RowCount-1]:= '1.0';
      StringGrid1.Cells[7,StringGrid1.RowCount-1]:=
        StringGrid1.Cells[7,StringGrid1.RowCount-2];
      StringGrid1.Cells[8,StringGrid1.RowCount-1]:=
        StringGrid1.Cells[8,StringGrid1.RowCount-2];
      StringGrid1.Col := 1;
      Exit;
    end;
    if key = vk_ReTurn then
      if (Trim(StringGrid1.Cells[0,x])='')and(Trim(StringGrid1.Cells[1,x])<>'') then
      begin
        Ss := True//ss为一个全局变量,防止按Enter键执行表格的OnSetEditText事件中的相关
        //代码
        DataSource1.DataSet := nil;
        DBGrid1.Visible := False;
        SelectSPXX;
        if DataModule1.ADOQuery1.RecordCount>1 then
        begin
          DataSource1.DataSet := DataModule1.ADOQuery1;
          DBGrid1.Visible := True;
          DBGrid1.SetFocus;
        end
        else if DataModule1.ADOQuery1.RecordCount = 1   then
        begin
          StringGrid1.Cells[0,x]:= DataModule1.ADOQuery1.FieldByName('商品编号').Value;
          StringGrid1.Cells[1,x]:= DataModule1.ADOQuery1.FieldByName('商品名称').Value;
          StringGrid1.SetFocus;
          StringGrid1.Col := 2;
        end
        else
        begin
          Application.MessageBox('该商品不存在。','提示',64);
          ClearString;
          if StringGrid1.RowCount>2 then
            StringGrid1.RowCount := StringGrid1.RowCount-1;
        end;
      end;
      if (Key = VK_Shift)and(JCxmlb = True) then
        BitBtn4.SetFocus;
  end;
```

16.8.2 配送审核管理设计

1. 概述

配送审核是管理人员对客户配送信息进行审批，确定是否允许送货。如果审批没有通过，不允许配送商品。配送审核运行结果如图 16.20 所示。



图 16.20 配送审核运行结果

2. 开发过程

（1）创建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TDataSource、TDBGrid、TCheckBox 等组件。

（2）设置各主要组件属性如表 16.7 所示。

表 16.7 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form31	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
DBGrid1	TDBGrid	DataSource	DataSource1
BitBtn1	TBitBtn	Enabled	False

3. 程序相关代码

自定义过程 Clear 主要完成程序初始化工作。

```
procedure TForm31.Clear;
begin
  //进行界面初始化
  ComboBox1.Text :=  '';
  Edit1.Clear;
  ComboBox2.ItemIndex := 0;
  DataModule1.ADOQuery2.Close;
end;
```

在窗体显示时向 ComboBox1 中添加配送票号，方便用户查询数据，用户查询数据时，

可以在 ComboBox1 中输入配送票号，也可以在 ComboBox1 中选择配送票号。

```
procedure TForm31.FormShow(Sender: TObject);
begin
  Combobox1.Clear;
  //向ComboBox1中添加配送票号
  begin
    with DataModule1.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 配送票号表  where (是否可用 = 0)and(完成送货 = 0) ');
      Open;
    end;
    if DataModule1.ADOQuery1.RecordCount >0 then
      while Not DataModule1.ADOQuery1.Eof do
      begin
        ComboBox1.Items.Add(DataModule1.ADOQuery1.FieldByName('配送票号').Value);
        DataModule1.ADOQuery1.Next;
      end;
  end;
  Clear;
  BitBtn1.Enabled := False;
  ComboBox1.SetFocus;
  CheckBox1.Checked := False;
end;
```

下面是“查询”的单击事件，单击“查询”按钮会将配送信息显示在列表中。在查询配送信息时，只查询没有审核的配送信息（通过“是否可用”字段标识，为 0 表示没有通过审核）。

```
procedure TForm31.SpeedButton1Click(Sender: TObject);
begin
  if Trim(ComboBox1.Text)<>''then
  begin
    CheckBox1.Checked := False;
    with DataModule1.ADOQuery2 do
    begin
      Close;//关闭数据集
      SQL.Clear;//清空SQL语句
      SQL.Add('select * from 配送申请表  where 配送票号 = (Select 配送票号 From 配送票号表
      where 配送票号 = :a and 是否可用 = 0)');
      //为参数赋值
      Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
      Open;//执行SQL语句
    end;
    if DataModule1.ADOQuery2.RecordCount>0 then //结果集是否有记录返回
    begin
```

```
        DataSource1.DataSet := DataModule1.ADOQuery2;
        BitBtn1.Enabled := True;
        Edit1.SetFocus;
      end
      else
      begin
        Application.MessageBox('该票号不存在或审核已经通过。','提示',64);
        ComboBox1.Text:= '';
      end;
    end;
end;
```

以下是“确定”按钮的单击事件，单击“确定”按钮会对当前配送信息进行审核。配送审核主要修改“配送票号表”，设置“审核负责人”，并将审核标识“是否可用”设置为0或1（根据是否通过审核）。

```
procedure TForm31.BitBtn1Click(Sender: TObject);
begin
  if Trim(Edit1.Text)='' then //审核人不能为空
  begin
    Application.MessageBox('请输入审核人。','提示',0+64);
    Edit1.SetFocus;
    Exit;
  end;
  Try
    with DataModule1.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('update 配送票号表 set 审核负责人 = :a,是否可用 = :c   where 配送票号 = :b');
      Parameters.ParamByName('a').Value := Trim(Edit1.Text);
      if Trim(ComboBox2.Text)= '允许' then
        Parameters.ParamByName('c').Value := 1
      else
        Parameters.ParamByName('c').Value :=0 ;
      Parameters.ParamByName('b').Value := Trim(DataModule1.ADOQuery2.FieldByName('配送票
      号').Value);
      ExecSQL;
    end;
    Application.MessageBox('审核完成。','提示',0+64);
    Self.OnShow(Sender);
  Except
    Application.MessageBox('系统出错。','提示',0+64);
    Close;
  end;
  BitBtn1.Enabled := False;
  end;
```

16.8.3 配送开票管理设计

1. 概述

配送开票是配送管理中最后一道程序，它是在商品出库后进行库存修改，同时，与客户进行结款。配送开票运行结果如图 16.21 所示。

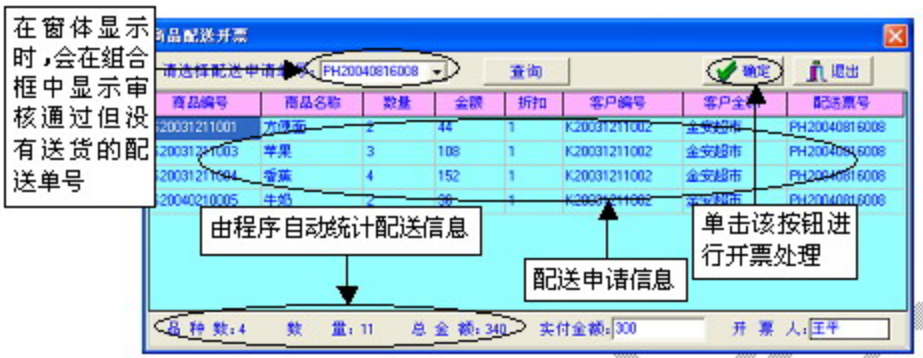


图 16.21 配送开票运行结果

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TStringGrid 等组件。

（2）设置各主要组件属性如表 16.8 所示。

表 16.8 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form32	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

在窗体显示时向 ComboBox1 中添加配送票号，调用 Clear 过程初始化界面。需要注意的是在向 ComboBox1 中添加配送票号时，这些票号是通过审核但没有进行配送的票号，程序通过“是否可用”字段（为 1 表示通过审核）和“完成送货”字段（为 0 表示没有进行送货）标识。

```
procedure TForm32.FormShow(Sender: TObject);
begin
  Czy = czymc;
  ComboBox1.Clear;
  with DataModule1.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 配送票号表 where (是否可用 =1)and(完成送货 =0)');
```

```
    Open;
  end;
  if DataModule1.ADOQuery1.RecordCount >0 then
    while Not DataModule1.ADOQuery1.Eof do
    begin
      //向ComboBox1中添加配送票号
      ComboBox1.Items.Add(DataModule1.ADOQuery1.FieldByName('配送票号').Value);
      DataModule1.ADOQuery1.Next;
    end;
  Clear;
  BitBtn1.Enabled := False;
end;
```

单击“查询”按钮查询配送信息。在进行查询时，只查询通过审核而没有进行配送的票号信息，程序通过“是否可用”字段（为 1 表示通过审核）和“完成送货”字段（为 0 表示没有进行送货）标识。

```
procedure TForm32.SpeedButton1Click(Sender: TObject);
var
  i: Integer;
begin
  ClearString;
  i:=1;
  if Trim(ComboBox1.Text)<>''then
  begin
    with DataModule1.ADOQuery2 do
    begin
      Close;//关闭数据集
      SQL.Clear;//清空SQL语句
      //添加SQL语句
      SQL.Add('select * from 配送申请表 where 配送票号 = (Select 配送票号 From 配送票号表
      where 配送票号 = :a and 是否可用 = 1 and 完成送货 = 0)');
      Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
      Open;//执行SQL语句，打开数据集
    end;
    if DataModule1.ADOQuery2.RecordCount>0 then
    begin
      while Not DataModule1.ADOQuery2.Eof do
      begin
        StringGrid1.Cells[0,i]:=  DataModule1.ADOQuery2.FieldByName('商品编号').Value;
        StringGrid1.Cells[1,i]:=  DataModule1.ADOQuery2.FieldByName('商品名称').Value;
        StringGrid1.Cells[2,i]:= IntToStr(DataModule1.ADOQuery2.FieldByName('数量').Value);
        StringGrid1.Cells[3,i]:= FloatToStr(DataModule1.ADOQuery2.FieldByName('金额').Value);
        StringGrid1.Cells[4,i]:= FloatToStr(DataModule1.ADOQuery2.FieldByName('折扣').Value);
        StringGrid1.Cells[5,i]:=  DataModule1.ADOQuery2.FieldByName('客户编号').Value;
        StringGrid1.Cells[6,i]:=  DataModule1.ADOQuery2.FieldByName('客户全称').Value;
        StringGrid1.Cells[7,i]:=  DataModule1.ADOQuery2.FieldByName('配送票号').Value;
        StringGrid1.RowCount := StringGrid1.RowCount +1;
```

```
            inc(i);
            DataModule1.ADOQuery2.Next;
          end;
          StringGrid1.RowCount := StringGrid1.RowCount-1;//删除最后的空行
          with DataModule1.ADOQuery1 do
          begin
            Close;
            SQL.Clear;
            SQL.Add('select * from 配送票号表 where 配送票号 = :a');
            Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
            Open;
          end;
          Label2.Caption := IntToStr(DataModule1.ADOQuery1.FieldByName('品种数').Value);
          Label4.Caption := IntToStr(DataModule1.ADOQuery1.FieldByName('数量').Value);
          Label5.Caption := FloatToStr(DataModule1.ADOQuery1.FieldByName('应付金额').Value);
          BitBtn1.Enabled := True;
          Edit2.SetFocus;
        end
        else
        begin
          Application.MessageBox('该票号不存在或不可用。','提示',64);
          ComboBox1.Text := '';
        end;
      end;
    end;
```

单击“确定”按钮完成开票处理。在开票过程中，需要修改库存信息，配送票号信息，同时，还需记录客户结款信息。

在开票过程中，首先检查基本信息是否为空，例如是否输入了审核人和实付金额。然后利用事务完成修改库存信息、配送票号信息、记录客户结款信息等操作，以保证数据的完整性。在修改库存时，如果库存数量不足，即库存数量小于配送数量，将进行提示并退出开票操作。

```
procedure TForm32.BitBtn1Click(Sender: TObject);
var
  jinja: Real; //商品进价，用来修改库存金额
  i: Integer;
  sum: Integer;
begin
  jinja:= 0.0;
  if Trim(Edit1.Text)='then
  begin
    Application.MessageBox('请输入审核人。','提示',0+64);
    Edit1.SetFocus;
    Exit;
  end;
  if Trim(Edit2.Text)='then
  begin
    Application.MessageBox('请输入实付金额。','提示',0+64);
```

```
    Edit2.SetFocus;
    Exit;
  end;
  if JCxmlb = False then
  begin
    DataModule1.ADOConnection1.BeginTrans; //开始一个事务
    Try   //捕捉异常
      For i := 1 to StringGrid1.RowCount-1 do
      begin
        Sum := GroupSL(Trim(StringGrid1.Cells[0,i]));
        with DataModule1.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('select * from 库存表 where 商品编号 = :a');
          Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,i]);
          Open;
        end;
        //如果配送商品数量大于库存数量，将不能完成送货
        if Sum > DataModule1.ADOQuery1.FieldByName('库存数量').Value then
        begin
          Application.MessageBox(Pchar('商品编号为'+StringGrid1.Cells[0,i]+的商品数量大于库存
          数量， 不能完成该操作。'),'提示',0+64);
          Exit;
        end;
      end;
      For i := 1 to StringGrid1.RowCount-1 do
      begin
        with DataModule1.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('select * from 商品基础信息表 where 商品编号 = :a');
          Parameters.ParamByName('a').Value := Trim(DataModule1.ADOQuery2.FieldByName('商
          品编号').Value);
          Open;
        end;
        jinja := DataModule1.ADOQuery1.FieldByName('进价').Value;
        with DataModule1.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('update 库存表 set 库存数量 = 库存数量 - :a, 库存金额= 库存金额 -:b where
          商品编号 = :c');
          Parameters.ParamByName('a').Value := StrToInt(StringGrid1.Cells[2,i]);
          Parameters.ParamByName('b').Value := jinja * StrToInt(StringGrid1.Cells[2,i]);
          Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[0,i]);
```

```
      ExecSQL;
    end;
end;
with DataModule1.ADOQuery1 do
begin
    Close;
    SQL.Clear;
    SQL.Add('update 配送票号表 set 开票人 = :a,完成送货=1   where 配送票号 = :b');
    Parameters.ParamByName('a').Value := Trim(Edit1.Text);
    Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[7,1]);
    ExecSQL;
end;
with DataModule1.ADOQuery1 do
begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 客户结款表 where 客户编号 = :a ');
    Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[5,1]);
    Open;
end;
if DataModule1.ADOQuery1.RecordCount >0 then
begin
    with DataModule1.ADOQuery1 do
    begin
        Close;
        SQL.Clear;
        SQL.Add('update 客户结款表 set 付款合计 = 付款合计 +:a ,欠款记账 = 欠款记账
        + :b');
        Parameters.ParamByName('a').Value := StrToFloat(Edit2.Text);
        Parameters.ParamByName('b').Value :=
        StrToFloat(Label5.Caption)-StrToFloat(Edit2.Text);
        ExecSQL;
    end;
end
else
begin
    with DataModule1.ADOQuery1 do
    begin
        Close;
        SQL.Clear;
        SQL.Add('insert 客户结款表 values (:a,:b,:c,:d,0.0,Default)');
        Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[5,1]);
        Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[6,1]);
        Parameters.ParamByName('c').Value := StrToFloat(Label5.Caption);
        Parameters.ParamByName('d').Value := StrToFloat(Label5.Caption);
        ExecSQL;
    end;
```

```
        end;
        with DataModule1.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('insert 客户结款记录表 values(:a,:b,:c,Default,:d)');
          Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[5,1]);
          Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[6,1]);
          Parameters.ParamByName('c').Value := StrToFloat(Edit2.Text);
          Parameters.ParamByName('d').Value := Trim(Czy);
          ExecSQL;
        end;
        DataModule1.ADOConnection1.CommitTrans;//提交事务
        Application.MessageBox('开票完成。','提示',0+64);
        Self.OnShow(Sender);//调用窗体的OnShow事件代码
        ComboBox1.SetFocus;
      Except //处理异常
        DataModule1.ADOConnection1.RollbackTrans;//事务回滚
        Application.MessageBox('系统出错。','提示',0+64);
        Close;
      end;
      BitBtn1.Enabled := False;
    end
    else
      Application.MessageBox('项目列表有误。','提示',64);
end;
```

16.8.4 配送查询管理设计

1. 概述

商品配送查询主要实现一段时间内对商品配送情况的查询。查询信息包括商品名称、数量、金额以及客户相应信息。配送查询运行结果如图 16.22 所示。



图 16.22 配送查询运行结果

2. 设计步骤

(1) 新建一个窗体，在窗体上放置 TPanel、TBitBtn、TSpeedButton、TLabel、TCheckBox、

TDataTimePicker、TDataSource、TDBGrid等组件。

（2）设置各主要组件属性如表16.9所示。

表16.9　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form56	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
CheckBox1	TCheckBox	Checked	False
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

防止在ComboBox1中输入字符。

```
procedure TForm56.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
  Key := #0;
end;
```

处理ComboBox1的OnChange事件，当ComboBox1中的内容改变时，ComboBox2中的内容会跟着相应地改变。

```
procedure TForm56.ComboBox1Change(Sender: TObject);
begin
  if Trim(ComboBox1.Text)<>''then
  begin
    with DataModule1.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select distinct');
      SQL.Add(Trim(ComboBox1.Text));
      SQL.Add('from 配送申请表 where 配送票号in(Select 配送票号 From 配送票号表where完
      成送货 = 1)');
      Open;
    end;
    ComboBox2.Clear;
    while Not DataModule1.ADOQuery1.Eof do //利用循环遍历结果集，将数据添加到组合框中
    begin
      ComboBox2.Items.Add(DataModule1.ADOQuery1.Fields[0].value);
      DataModule1 .ADOQuery1.Next;
    end;
  end;
end;
```

以下是“查询”按钮的单击事件，单击“查询”按钮，程序能够根据用户设置的条件进行动态查询。在查询时，首先检查查询条件是否为空，即ComboBox1与ComboBox2中是否有数据，如果为空进行提示并退出查询，否则进一步判断是否按时间段查询

（CheckBox2 是否被选中），如果按时间段查询，还需要检查时间段是否合法，即起始时间是否大于终止时间，如果时间段不合法，进行提示并退出查询。

```
procedure TForm56.SpeedButton1Click(Sender: TObject);
begin
  if (Trim(ComboBox2.Text)<>'')and(Trim(ComboBox1.Text)<>'') then
  begin
    CheckBox1.Checked := False;
    with DataModule1.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      if CheckBox2.Checked = False then
      begin
        SQL.Add('select * from 配送申请表 where ');
        SQL.Add(Trim(ComboBox1.Text));
        SQL.Add('=');
        SQL.Add(':a');
        SQL.Add('and 配送票号 in (Select 配送票号 From 配送票号表 where 完成送货 = 1)');
      end
      else
      begin
        SQL.Add('select * from 配送申请表 where ');
        SQL.Add(Trim(ComboBox1.Text));
        SQL.Add('=');
        SQL.Add(':a');
        SQL.Add('and 配送票号 in (Select 配送票号From 配送票号表where 完成送货 = 1 and (配
        送时间 >= :b)and(配送时间 <= :c))');
        Parameters.ParamByName('b').Value := Int(DateTimePicker1.Date)-2;//将时间取整是为
        //了从起始日期的零时开始计算，减2（如果不是嵌套的Select语句，不需要减2）是为了
        //与SQL Server中的时间对应
        Parameters.ParamByName('c').Value := Int(DateTimePicker2.Date)-2+0.9998;
      end;
      Parameters.ParamByName('a').value := Trim(ComboBox2.Text);
      Open;
    end;
    if DataModule1 .ADOQuery2.RecordCount>0 then
    DataSource1.DataSet := DataModule1 .ADOQuery2;
  end;
end;
```

16.8.5 系统登录程序设计

1. 概述

在开发应用程序时，为提高系统的安全性，一般都设置系统登录。在程序运行时，首先显示登录界面，要求用户输入操作员名称和密码，只有操作员名称和密码都正确才能进入系统。系统登录运行结果如图 16.23 所示。

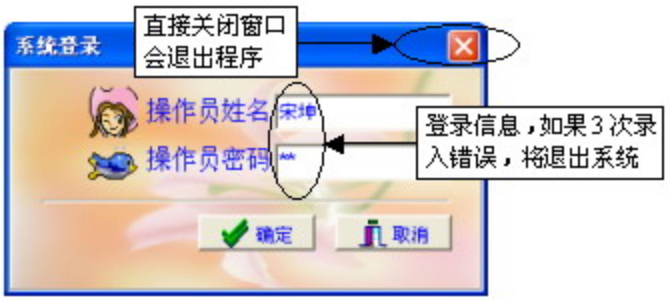


图 16.23　系统登录

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TImage、TLabel、TEdit 等组件。

（2）设置各主要组件属性如表 16.10 所示。

表 16.10　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Forma	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
Edit2	TEdit	PasswordChar	*
BitBtn1	TBitBtn	Enabled	False

3. 程序相关代码

下面代码主要判断“确定”按钮是否可用，Edit2 的 OnChange 事件也调用了该方法。如果操作员姓名或操作密码为空，“确定”按钮不可用。

```
procedure TForma.Edit1Change(Sender: TObject);
begin
  if (Trim(Edit1.Text)<>'')and( Trim(edit2.Text)<>'') then
    BitBtn1.Enabled := True
  else
    BitBtn1.Enabled := False;
end;
```

下面是“确定”按钮的单击事件，用于验证用户身份，只有合法的用户才能进入系统。程序以操作员姓名和密码为条件从操作员信息表中查询数据，如果有数据返回，表明用户身份合法，可以进入系统，反之，用户不合法，累计次数（如果累计次数超过 3 次将退出系统）。

```
procedure TForma.BitBtn1Click(Sender: TObject);
begin
  with DataModule1.ADOQuery1 do //以操作员名称和密码为条件从操作员信息表中查询数据，如果
  //有数据返回，表明用户输入正确，否则不正确
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 操作员信息表 where (操作员姓名= :a)and(操作员密码= :b)');
    Parameters.ParamByName('a').Value := Trim(Edit1.Text);
```

```
      Parameters.ParamByName('b').Value := Trim(Edit2.Text);
      Open;
    end;
    if DataModule1.ADOQuery1.RecordCount >0 then
    begin
      czymc := DataModule1.ADOQuery1.FieldByName('操作员姓名').AsString;
      m:= True;
      Form90.StatusBar1.Panels[3].Text := czymc;
      Edit1.Clear;
      Edit2.Clear;
      Close;
    end
    else
    begin
      if i <3 then
      begin
        Application.MessageBox('用户名称或密码不正确，请重新输入。','提示',0+64);
        Edit1.Clear;
        Edit2.Clear;
        Edit1.SetFocus;
        Inc(i);
      end
      else
      begin
        Application.MessageBox('对不起，您无权使用本系统。','提示',0+64);
        Application.Terminate ;
      end;
    end;
  end;
```

以下是窗体关闭事件，如果用户没有登录，直接关闭窗口，应用程序将结束。如果不写该段代码，系统登录将形同虚设，不具有任何意义，因为如果不写该段代码，在程序运行时关闭登录界面，会直接进入主程序。

```
  procedure TForma.FormClose(Sender: TObject; var Action: TCloseAction);
  begin
    if m = False then
      Application.Terminate;
  end;
```

16.9 疑难问题分析解决

16.9.1 如何在 TImageList 中存储大图标

在使用 TImageList 存储图标时，TImageList 默认的宽度和高度为 16，存储的图标比较小，可以将 TImageList 宽度和高度调整为 48 或者 64，这样，TImageList 所存储的图标就会比较大。

需要注意的是，当调整 TImageList 的宽度和高度时，TImageList 原先所存储的图表会被清空。

16.9.2 如何实现动态查询

在本系统的配送查询模块中，用户在 ComboBox1 中选择不同的字段时，在 ComboBox2 中会显示相应的字段值。在组合框控件 TComboBox 组件中提供了 OnChange 事件，该事件在组合框中的当前选项发生变化时触发。用户只要在该事件中编写代码，便可以实现动态查询。示例如图 16.24 所示。

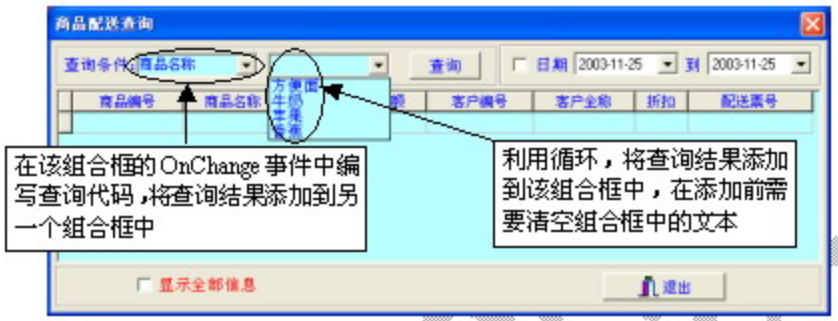


图 16.24 动态查询

代码如下：

```
with DataModule1.ADOQuery1 do
begin
  Close;
  SQL.Clear;
  SQL.Add('select distinct'); //Destinct 关键字用于删除重复数据
  SQL.Add(Trim(ComboBox1.Text));//ComboBox1 中的数据为用户选择的字段名称
  SQL.Add('from 配送申请表 where 配送票号 in (Select 配送票号 From 配送票号表 where 完
      成送货 = 1)');
  Open;
end;
ComboBox2.Clear; //注意，一定要清空ComboBox2中的数据
while Not DataModule1.ADOQuery1.Eof do
begin
  ComboBox2.Items.Add(DataModule1.ADOQuery1.Fields[0].value);
  DataModule1 .ADOQuery1.Next;
end;
```

16.9.3 如何去除重复行

用户在使用程序时，可能会向数据表中录入许多重复的数据。例如在录入客户信息时，可能几个客户信息的客户全称相同，当我们从客户基础信息表中查询客户全称时，可能会将所有的客户全称都查询出来，其中一些重复的数据是多余的，需要将其清除。在 SQL Server 中，提供了“Distinct”关键字来去除重复行，示例如下：

```
with DataModule1.ADOQuery1 do
begin
  Close;//关闭数据集
```

```
    SQL.Clear;//清空SQL语句
    SQL.Add('select distinct 客户全称 from 客户基础信息表');//添加SQL语句
    Open;//打开数据集
end;
```

16.9.4 动态创建窗体

在开发程序时，由于一个应用程序会包含许多窗体，如果每个窗体都静态地创建，程序在运行时会减慢，同时，耗费大量的内存资源。所以，通常情况下，除主窗体和一些特殊窗体（数据模块或被子窗体继承的窗体等）外，其他窗体都应动态创建。实现动态创建窗体，可以在设计完一个窗体后，单击“Project”/“Options”菜单项，打开工程选项对话框，如图 16.25 所示。

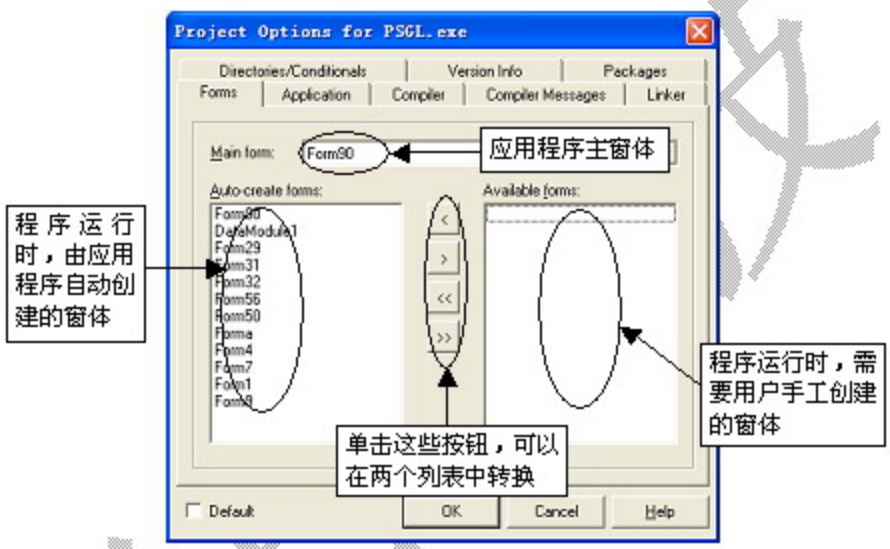


图 16.25 工程选项对话框

在“Auto-create forms”列表中选中可以动态创建的窗体，单击“>”按钮，将其移入“Available forms”列表中，如图 16.26 所示。

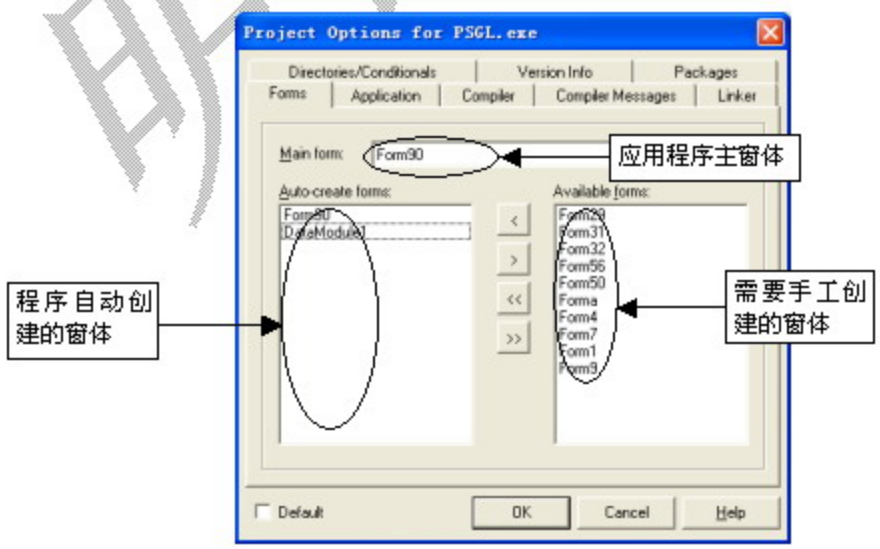


图 16.26 动态创建窗体

“Available forms”列表中的窗体在使用前，需要创建，否则会出现地址访问错误。例如，下面的程序代码在执行时，会弹出如图 16.27 所示的错误。

```
procedure TForm90.N4Click(Sender: TObject);
begin
  Form7.ShowModal;
  Form7.Free;
end;
```

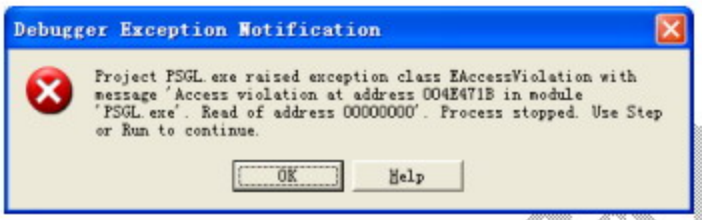


图 16.27　地址访问错误

由于“Form7”在使用前没有被创建，而直接调用它的方法，当然会出现错误。因此，上面的程序代码应该为：

```
procedure TForm90.N4Click(Sender: TObject);
begin
  Application.CreateForm(TForm7,Form7);//创建窗体
  Form7.ShowModal;
  Form7.Free;
end;
```

16.9.5　如何实现配送开票查询

在配送开票时，需要根据配送票号查询配送信息。但是，该配送票号必须是审核通过的配送票号，而且没有完成送货。在配送票号表中有两个字段分别标识配送票号是否通过审核，配送票号是否完成送货。在配送开票模块，为了方便用户查询，在窗体显示时会将“符合条件”的配送票号添加到组合框中，如图 16.28 所示。

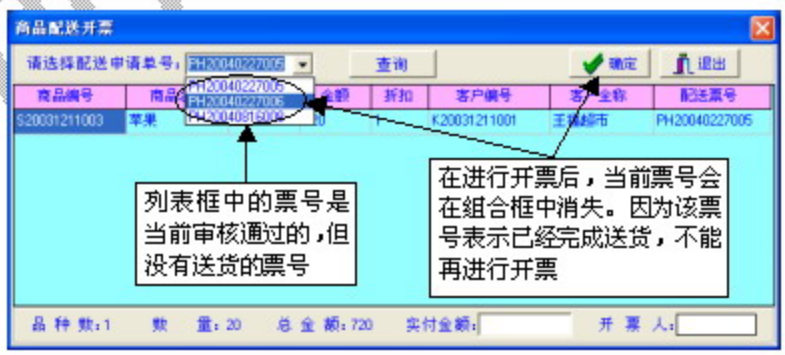


图 16.28　配送开票查询

代码如下：

```
with DataModule1.ADOQuery1 do
begin
```

```
Close;
  SQL.Clear;
  //字段“是否可用”为1表示审核已经通过，字段“完成送货”为0表示还没有送货
  SQL.Add('select * from 配送票号表 where (是否可用 = 1)and(完成送货 = 0)');
  Open;
end;
if DataModule1.ADOQuery1.RecordCount >0 then
  while Not DataModule1.ADOQuery1.Eof do
  begin
    ComboBox1.Items.Add(DataModule1.ADOQuery1.FieldByName('配送票号').Value);
    DataModule1.ADOQuery1.Next;
  end;
```

16.9.6 如何将表格中某些列设置为只读

在利用表格录入数据时，对于用户录入过的某些数据要求不允许修改。例如，在配送申请管理模块中，如果商品编号存在，则不允许再修改商品名称。在 TStringGrid 组件中，提供了 OnSelectCell 事件，该事件在表格中单元格被选中时触发。在单元格被选中时，可以通过该事件的参数确定当前被选中单元格的横、纵坐标。用户在录入商品名称时，程序会判断当前行商品编号是否存在，如果存在会将表格设置为不可编辑状态，如图 16.29 所示。

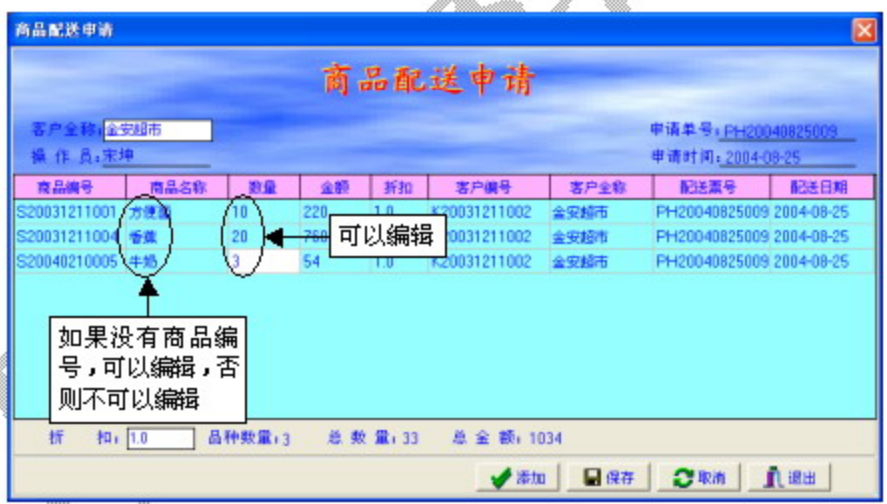


图 16.29 将表格字段设置为只读

代码如下：

```
procedure TForm29.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if (ACol = 1)and( Trim(StringGrid1.Cells[0,x])='')or(ACol =
    2)and(Trim(StringGrid1.Cells[0,x])<>'') then
    StringGrid1.Options := StringGrid1.Options +[goEditing]
  else
    StringGrid1.Options := StringGrid1.Options -[goEditing];
end;
```

16.10 程序调试及错误处理

16.10.1 程序调试

在开发应用程序时，不可避免会出现这样或那样的错误，有些错误是在设计或编译时产生的，这类错误只能由开发人员发现，有些错误是在运行时产生的，程序开发人员和用户都可以发现这类错误。所以，一个应用程序的完成往往需要经过测试、修改、调试、再测试等步骤，如此反复重复，逐步完善应用程序。

在 Delphi 中提供了许多调试手段供开发人员使用，都在“Run”菜单下。下面将举例说明如何使用 Delphi 提供的调试工具调试程序。

下面程序代码是最初配送审核窗口“查询”按钮的单击事件代码：

```
procedure TForm31.SpeedButton1Click(Sender: TObject);
begin
  if Trim(ComboBox1.Text)<>''then
  begin
    CheckBox1.Checked := False;
    with DataModule1.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 配送申请 where 配送票号 =(Select 配送票号 From 配送票号表
      where 配送票号 = :a and 是否可用 = 0)');
      Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
      Open;
    end;
    if DataModule1.ADOQuery2.RecordCount>0 then
    begin
      DataSource1.DataSet := DataModule1.ADOQuery2;
      BitBtn1.Enabled := True;
      Edit1.SetFocus;
    end
    else
    begin
      Application.MessageBox('该票号不存在或审核已经通过。','提示',64);
      ComboBox1.Text:= '';
    end;
  end;
end;
```

程序运行时，在配送审核窗口中单击“查询”按钮会出现如图 16.30 所示的错误提示。

由于问题出现在“查询”按钮的单击事件里，所以，需要对该段代码进行调试。首先将鼠标光标停在“查询”按钮单击事件的首行代码，单击“Run”/“Add BreakPoint”/“Source BreakPoint”菜单项或按 F5 键设置断点，效果如图 16.31 所示。

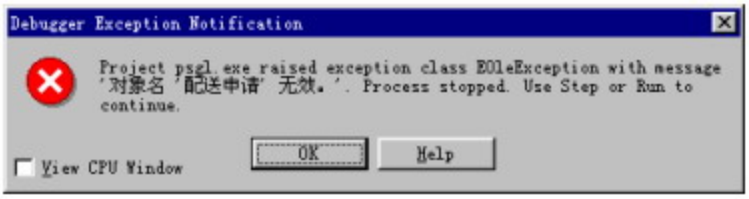


图 16.30　错误提示

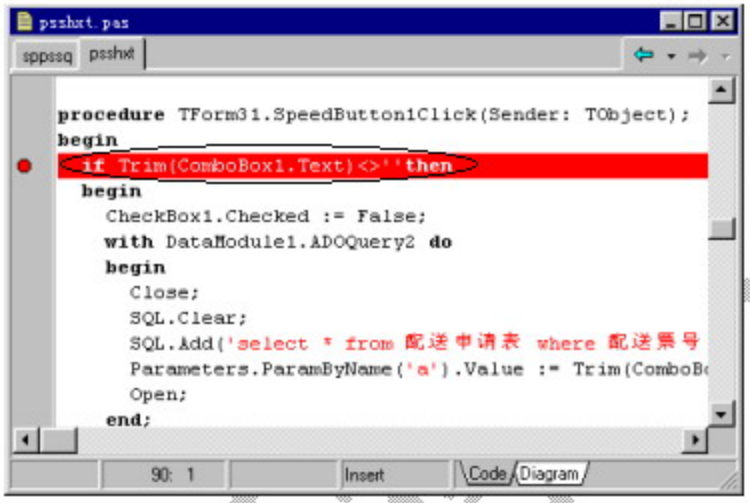


图 16.31　设置断点

运行程序，打开配送审核窗口，单击“查询”按钮，如图 16.32 所示。

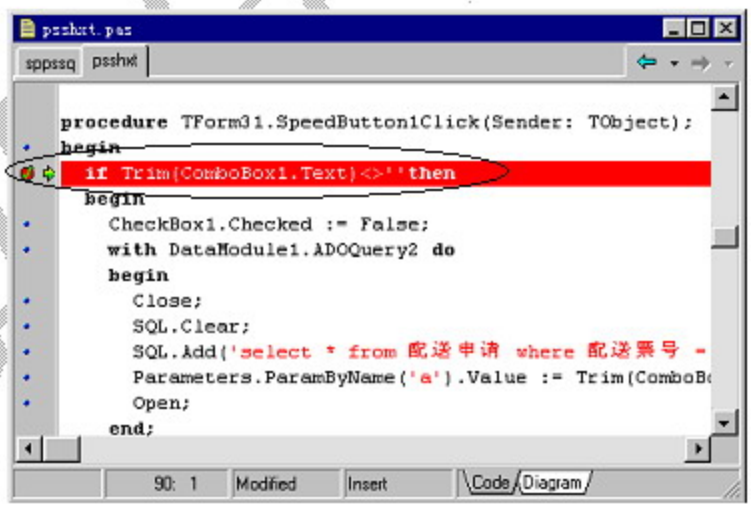


图 16.32　调试界面

按 F7 键单步执行程序，当程序执行完“Open”语句后会弹出图 16.30 所示的错误提示，可以断定是 SQL 语句有问题，因为“Open”语句的作用是返回查询结果集。错误提示是“对象名配送申请无效”，仔细检查 SQL 语句，会发现原来是数据表名称写错了，将 SQL 语句中的“配送申请”改为“配送申请表”，保存所做的修改，重新运行程序，错误消失了。

16.10.2 异常处理

异常处理是对程序中产生的异常或者错误进行处理，它是防止程序在执行时因产生异常而中断的一道防线。也许你认为所开发的程序是可以执行的，不需要异常处理，然而，能够通过编译的程序并不表示是完美的，有些异常是在程序执行时产生的，16.10.1 小节中的例子便是很好的证明，况且程序还会受软、硬件环境的影响，因此，在开发应用程序时，异常处理是不可少的。

在 Object Pascal 中提供了两种异常处理语句，分别为“Try…Except…End”和“Try…Finally…End”。下面将详细介绍这两种语句的用法。

- Try…Except…End

使用该语句可以让开发人员自行捕捉异常，然后根据异常的不同类型做不同的处理。请看下面的语法内容：

```
Try
    语句；//可能产生异常的语句
    ...
 Except
    语句；//处理异常的语句
    ...
 End;
```

如果“Try…Except”之间的语句没有产生异常，程序会跳过“Except…End”区内的程序代码，执行其后面的程序。如果“Try…Except”之间的语句产生异常，程序就会从产生异常的位置跳到“Except…End”区，执行“Except…End”之间的程序代码。请看下面的例子。

```
procedure TForm29.Edit8Exit(Sender: TObject);
begin
    try
      StrToFloat(Edit8.Text);
   except
      Application.MessageBox('请输入合法字符。','提示',0+64);
      Edit8.Clear;
      Edit8.SetFocus;
   end;
end;
```

如果用户在 Edit8 中输入字符等非实数，当焦点离开 Edit8 时便会产生异常，同时弹出提示对话框。

- Try…Finally…End

“Try…Finally”之间是可能发生异常的语句，“Finally…End”之间的程序代码无论是否有异常产生，都将会执行。请看下面的例子。

```
procedure TForm29.Edit8Exit(Sender: TObject);
begin
   try
      StrToFloat(Edit8.Text);
   Finally
```

```
        Application.MessageBox('请输入合法字符。','提示',0+64);
        Edit8.Clear;
        Edit8.SetFocus;
      end;
end;
```

无论用户在 Edit8 中是否输入实数，当焦点离开 Edit8 时，都会执行“Finally…End”之间的代码。

16.11 程序设计清单

配送管理系统文件清单如表 16.11 所示。

表 16.11　　程序设计清单

文 件 名	文 件 类 型	说　明
Data1.dfm	窗体文件	数据模块
Kcgl.dfm	窗体文件	仓库管理窗体
Khcx6.dfm	窗体文件	显示客户信息窗体
Khcxgl.dfm	窗体文件	客户基础信息窗体
Pskpxt.dfm	窗体文件	配送开票管理窗体
Psshxt.dfm	窗体文件	配送审核管理窗体
Pssqgl.dfm	窗体文件	配送申请管理窗体
Psjcxxgl.dfm	窗体文件	商品基础信息窗体
Sppscx.dfm	窗体文件	商品配送查询窗体
Sppssq.dfm	窗体文件	配送管理主窗体
Tjyusc.dfm	窗体文件	操作员管理窗体
Xtdl1.dfm	窗体文件	系统登录窗体
Psgl.dpr	工程文件	
Psgl.exe	可执行文件	

16.12 技术、经验总结

16.12.1 技术总结

在本系统的查询管理模块中，为了从配送申请表中查询已配送商品的配送信息，程序中使用了两层“Select”语句，请看下面的代码：

```
with DataModule1.ADOQuery2 do
begin
  Close;
  SQL.Clear;
  SQL.Add('select * from 配送申请表 where ');
```

```
    SQL.Add(Trim(ComboBox1.Text));
    SQL.Add('=');
    SQL.Add(':a');
    SQL.Add('and 配送票号 in(Select 配送票号 From 配送票号表 where 完成送货 = 1)');
    Parameters.ParamByName('a').value := Trim(ComboBox2.Text);
    Open;
  end;
```

需要注意的是，子查询的“Select”语句应使用圆括号括起来，并且在子查询中不能使用“COMPUTE”或“FOR BROWSE”关键字。

16.12.2 经验总结

在现实生活中，由于事物缤纷复杂，经常会出现重名现象，因此，程序员在设计数据表结构时，不能使用事物名称来区分事物，通常的做法是用事物编号来区分事物。在程序中为每一个事物添加一个惟一的事物编号。但是，当用户录入数据时，使用事物编号代替事物名称是不现实的，因为用户必须记住每一个事物编号所对应的事物名称。在程序中，用户录入事物名称时，由程序判断事物名称是否重名，如果重名，可以将所有重名的事物列举出来，让用户选择。

16.13 程序的打包与发行

在开发完一个应用程序后，一般需要对其进行打包，以便于安装到目标计算机中。在Delphi的安装盘中一般都提供了一个Install Shield限定版软件，使用该软件便可以对程序进行打包。下面以具体实例介绍如何使用Install Shield对程序进行打包。

（1）打开Install Shield程序，主界面如图16.33所示。

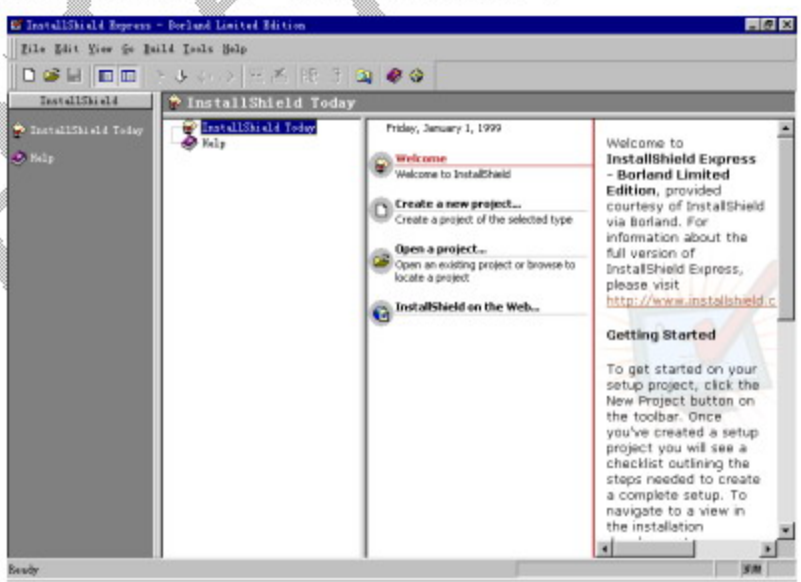


图16.33 Install Shield主界面

（2）选择“File”/“New”菜单项，打开如图16.34所示的对话框。在其中填入包的文件名或路径。本例为“C:\配送管理”。



图 16.34　工程对话框

（3）单击 OK 按钮，打开如图 16.35 所示的窗口。

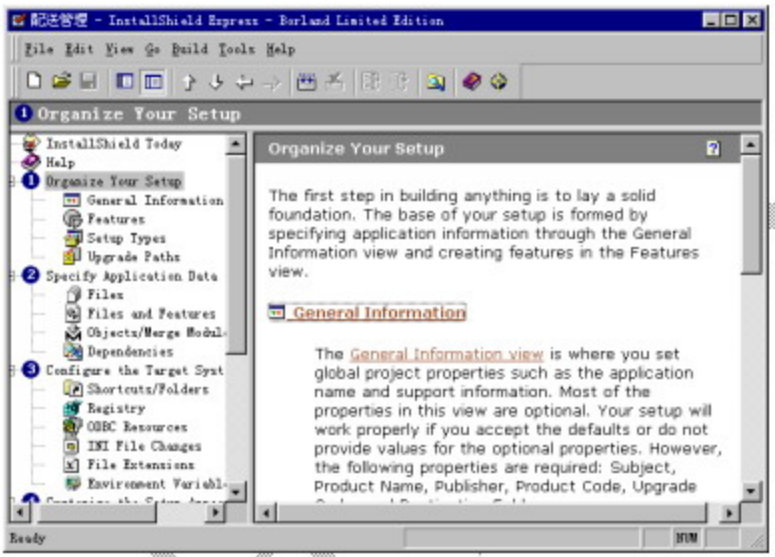


图 16.35　工程主界面

（4）选中“General Information”选项，在右边的属性列表中，设置“Product Name”属性为“配送管理系统”，“INSTALLDIR”属性为“D:\配送管理”。如图 16.36 所示。

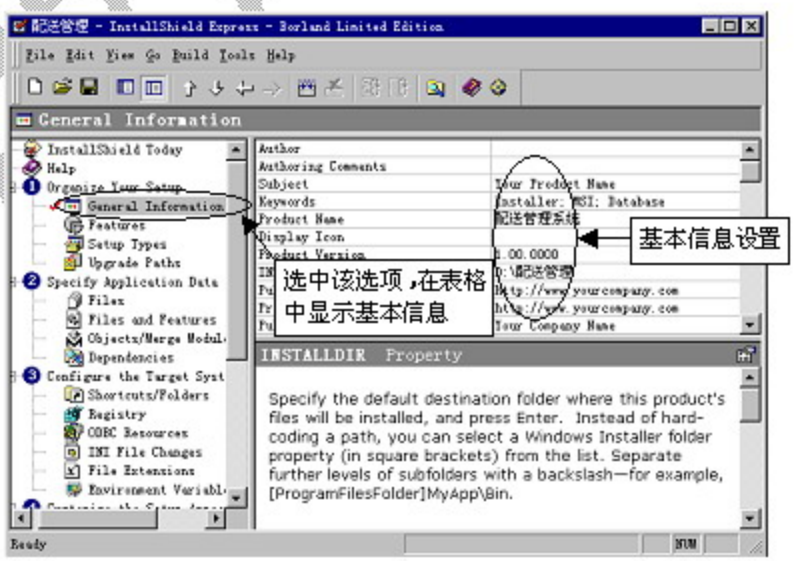


图 16.36　属性设置

（5）选中“Files”选项，用鼠标右击属性列表下方的“Destination Computer”选项。选择“Show Predefined Folder”/“INSTALLDIR”菜单，在“Destination Computer”选项下方将出现一个“INSTALLDIR”文件夹。如图 16.37 所示。

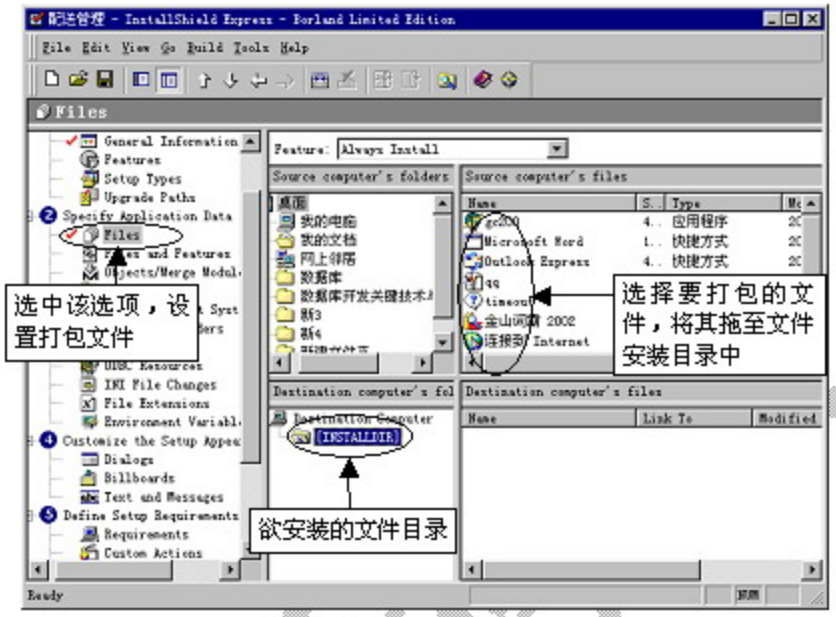


图 16.37　文件设置

（6）在“Source Computer's Folders”选项中选择要安装的文件。本例为“D:\进销存\配送管理”，选中所有文件，按住鼠标左键，将它拖至“INSTALLDIR”中。

（7）按 F7 键进行编译。至此完成安装程序的全部设置。

第 17 章 Delphi 7 在仓库管理系统中的应用

17.1 仓库管理系统概述

17.1.1 仓库管理系统简介

仓库管理系统是一套针对商品仓库与仓库存储商品等信息的管理及维护的管理软件。它能够辅助仓库管理人员对供应商信息、商品基本信息以及商品入库、出库、库存信息查询等进行控制管理。它在仓库管理中的应用不仅可以简化、规范商品信息管理的日常操作，而且可以使商品管理更加简单、方便、快捷、清晰，从而减轻工作人员的劳动强度，同时避免了人工统计管理的麻烦，杜绝各项信息在管理中的操作失误。计算机管理系统的应用是各企业、公司迈向现代化企业的重要标识，必将成为商品管理现代化不可缺少的手段。

17.1.2 系统功能描述

- 供应商基本信息录入
- 商品基本信息录入
- 商品入库管理录入
- 商品出库管理录入
- 商品库存查询
- 密码修改设置
- 数据库备份、恢复

17.2 系统调查

随着我国国民经济的快速发展，企业规模的不断壮大，由原来的手工管理逐步转变为信息化管理，计算机已不是企业的点缀，它在企业信息管理中占据举足轻重的位置。面对当前企业管理手段、管理技术落后等诸多问题，使用计算机管理在很大程度上可以帮助企业管理克服这些困难，计算机管理系统可以提高企业的管理水平，节省大量人力、物力和资源的消耗，使经营管理更加科学有序，经营物流更清晰、翔实、准确。本系统就是利用计算机信息技术，通过商品入库、出库等信息，对商品库存情况及时了解，为企业发展提供重要保证。

17.3 需求分析

1. 系统目标

使用计算机实现商品入库、出库管理，随时查询商品库存情况，取代人工记账。

2. 系统功能

- 供应商基本信息

在供应商基本信息中应包含供应商名称、供应商地址、联系人、联系电话、所属地区等信息。在供应商基本信息中需要实现供应商基本信息的添加、浏览、修改、删除。

- 商品基本信息

在商品基本信息中包含商品编号、商品名称、规格、产地、进价、批发价、零售价、保质期等基本信息。在商品基本信息中，根据供应商所提供的商品，进行商品详细信息的录入。在商品基本信息中需要实现商品基本信息的添加、浏览、修改、删除。

- 商品入库信息

在商品入库信息中，通过基本信息中的供应商编号，得到相对应的供应商名称，同时得到供应商所提供的商品。对商品进行入库操作。商品入库信息还包含入库编号、入库日期、经手人等信息。在商品入库信息中需要实现商品入库信息的添加、删除。

- 商品出库信息

在商品出库信息中，通过入库编号，获取商品名称、商品单位、商品单价等相关信息。通过这些基本信息，对商品进行出库操作。商品出库信息还包含出库编号、出库日期、经手人等信息。在商品出库信息中需要实现商品出库信息的添加、删除。

- 商品库存查询

在商品库存查询中通过商品名称对商品规格、商品单位、商品单价、商品数量进行查询。

- 密码修改设置

在密码修改设置中，通过原来的密码，对操作员及密码进行修改。

- 数据库备份、恢复

在数据库备份中，单击“数据备份”按钮，根据当前日期对数据库中的数据进行备份，并保存到指定的数据备份文件夹中。

在数据恢复中，单击“数据恢复”按钮，打开以前备份的数据库文件，并执行恢复操作。

明日科技

3. 系统结构图

仓库管理系统的结构图如图 17.1 所示。

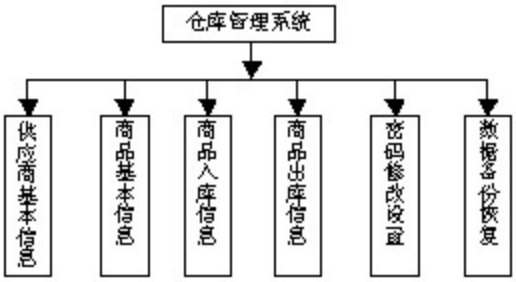


图 17.1　仓库管理系统结构图

17.4　系 统 设 计

17.4.1　系统流程图

仓库管理系统的流程图如图 17.2 所示。

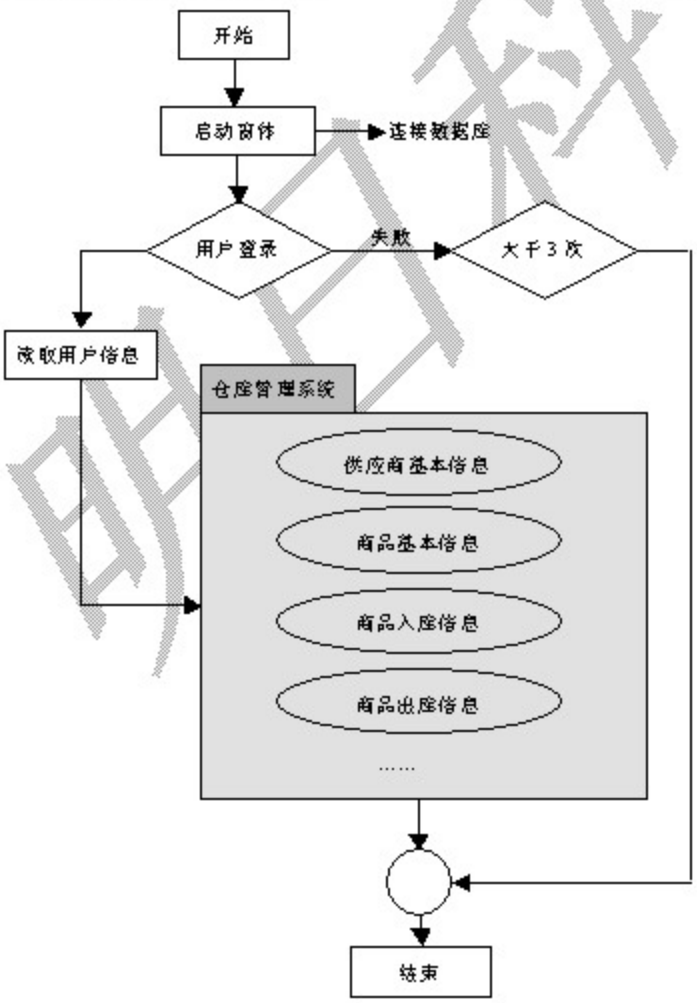


图 17.2　仓库管理系统流程图

17.4.2 业务流程

仓库管理系统的业务流程图如图 17.3 所示。

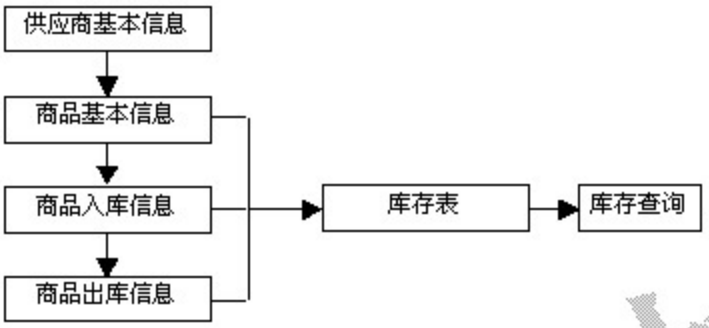


图 17.3 仓库管理系统业务流程图

17.5 数据库设计

17.5.1 数据库概念设计

仓库管理系统 E-R 图如图 17.4 所示。



图 17.4 系统 E-R 图

17.5.2 数据库逻辑结构设计

仓库管理系统共使用了 6 张数据表，分别为：供应商基本信息、商品基础信息、商品入库信息、商品出库信息、库存表、用户表。具体如表 17.1～表 17.6 所示。

表 17.1 用户表

字段名称	字段类型	主 键	外 键	是否为空	默 认 值
用户编号	Varchar(15)			否	
用户名称	Varchar(15)				
密码	Varchar(20)				

表 17.2 供应商基本信息表

字段名称	字段类型	主 键	外 键	是否为空	默 认 值
供应商编号	Varchar(30)			否	
供应商名称	Varchar(30)				
供应商地址	Varchar(40)				

清华科技

续表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
联系人	Varchar(12)				
联系电话	Varchar(12)				
传真	Varchar(12)				
邮政编码	Varchar(10)				
所属地区	Varchar(10)				
开户银行	Varchar(16)				
银行账号	Varchar(30)			否	
备注	Varchar(50)				

表 17.3　　商品基本信息表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
商品编号	Varchar(30)			否	
商品名称	Varchar(20)				
商品简称	Varchar(10)				
规格	Varchar(10)				
保质期	Varchar(10)				
产地	Varchar(30)				
单位	Varchar(10)				
商品单价	Float(8)				
批发价	Float(8)				
零售价	Float(8)			否	
供应商编号	Varchar(30)				
供应商名称	Varchar(30)				
备注	Varchar(50)				

表 17.4　　商品入库信息表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
供应商编号	Varchar(30)			否	
供应商名称	Varchar(30)				
入库编号	Varchar(30)				
商品名称	Varchar(20)				q
商品规格	Varchar(12)				
商品单位	Varchar(10)				
商品数量	Varchar(10)				
商品单价	Flaot(8)				
入库日期	DateTime(12)				
经手人	Varchar(12)				
备注	Varchar(50)				

表 17.5　　商品出库信息表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
出库编号	Varchar(30)			否	
商品编号	Varchar(30)				
商品名称	Varchar(20)				
商品单位	Varchar(10)				
商品数量	Varchar(10)				
商品单价	Float(8)				
出库日期	DateTime(8)				
经手人	Varchar(12)				
备注	Varchar(50)				

表 17.6　　库存表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
商品名称	Varchar(20)			否	
商品单位	Varchar(10)				
商品规格	Varchar(12)				

17.5.3　数据字典

表 17.7　　供应商基础信息表

字 段 名 称	描　　述	说　　明
供应商编号	标识一个供应商	用来区分同名的供应商
供应商名称	标识供应商名称	

表 17.8　　库存表

字 段 名 称	描　　述	说　　明
商品名称	标识一种商品	用来区分同名的商品
商品数量	记录商品数量	
商品单价	记录商品单价	

17.6　处理过程分析

17.6.1　商品入库处理过程分析

商品入库是一个从供应商进货的过程，在进货时需要记录供应商信息、商品入库信息，所以在商品入库时需要进行登记，记录从哪个供应商进了哪些商品，以及商品数量、金额等信息。在商品入库过程中，关键之处是修改商品库存，用户在保存信息时，程序除了检查用户录入信息是否有错误，还要修改商品库存，如果用户录入的商品在库存中不存在，则向库存中插入商品信息（商品名称、商品数量等），如果存在，则在库存中修改该商品库存数量。商品入库管理业务流程如图 17.5 所示。

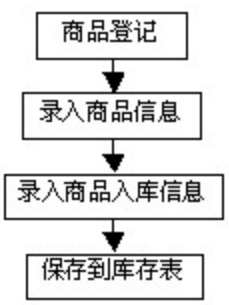


图 17.5　商品入库开发流程

17.6.2 商品出库处理过程分析

商品出库是一个提货的过程。在商品出库时，首先进行商品出库登记，然后录入提货的商品信息，最后修改库存，保存提货信息。商品出库业务流程如图 17.6 所示。



图 17.6 商品出库业务流程图

17.7 主窗体设计

主窗体设计步骤如下。

（1）打开 Delphi 7，创建一个含有窗体的工程，在该窗体上放置 TImage、TOpenDialog、TSaveDialog、TTimer、TLabel、TSpeedButton 等组件，如图 17.7 所示。



图 17.7 主程序界面

（2）设置各主要组件属性如表 17.9 所示。

表 17.9 组件属性设置

对 象 名	对 象 类 型	属 性	值
MAIN	TForm	Color	clInfoBk
Image1	TImage	AutoSize	True
SpeedButton1	TSpeedButton	Cusror	crHandPoint

		Flat	True

明日科技

17.8 数据模块设计

数据模块设计步骤如下。

（1）打开工程“ckgl”，单击“File”/“New”/“Data Module”菜单项，创建数据模块。

（2）选中数据模块，在对象观察器中修改“Name”属性为“DM”。

（3）在数据模块上放置 TADOConnection、TDataSource、TADOQuery、TADODataSet 组件，如图 17.8 所示。

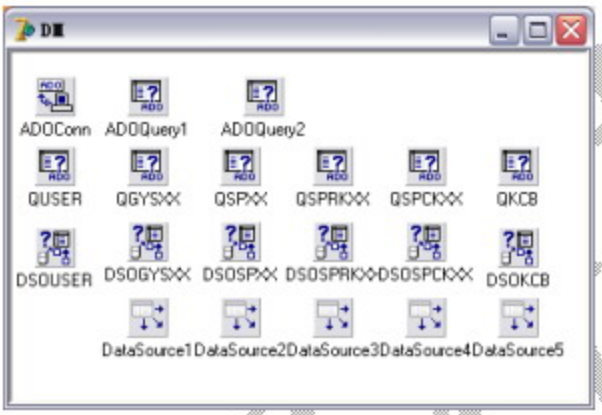


图 17.8 数据模块

（4）设置各主要组件属性如表 17.10 所示。

表 17.10 组件属性设置

对 象 名	对 象 类 型	属 性	值
ADOConn	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=CKGL
QUSER	TADOQuery	Connection	ADOConn
DSOUSER	TADODataSet	Connection	QUSER
QGYSXX	TADOQuery	Connection	ADOConn
DSOGYSXX	TADODataSet	Connection	QGYSXX
DataSource1	TDataSource	DataSet	DSOGYSXX
ADOQuery1	TADOQuery	Connection	ADOConn
ADOQuery2	TADOQuery	Connection	ADOConn

17.9 程序设计与编码

17.9.1 商品入库管理设计

1. 概述

商品入库管理主要实现商品的入库登记。在商品入库时，需要记录商品数量、商品金额、

商品种类等信息，同时，需要修改商品库存数量。在本系统的商品入库管理中，用户单击“添加”按钮，程序将生成入库编号，在选择供应商编号之后，在供应商名称中显示供应商名称后按〈Enter〉键，焦点移到商品名称中，选择商品名称，会将商品信息显示在其他编辑框中。输入商品数量、入库日期、经手人，单击“保存”按钮。将商品入库信息显示在“商品入库信息主窗体”中。商品入库编辑窗体、商品入库信息主窗体运行结果如图 17.9、17.10 所示。

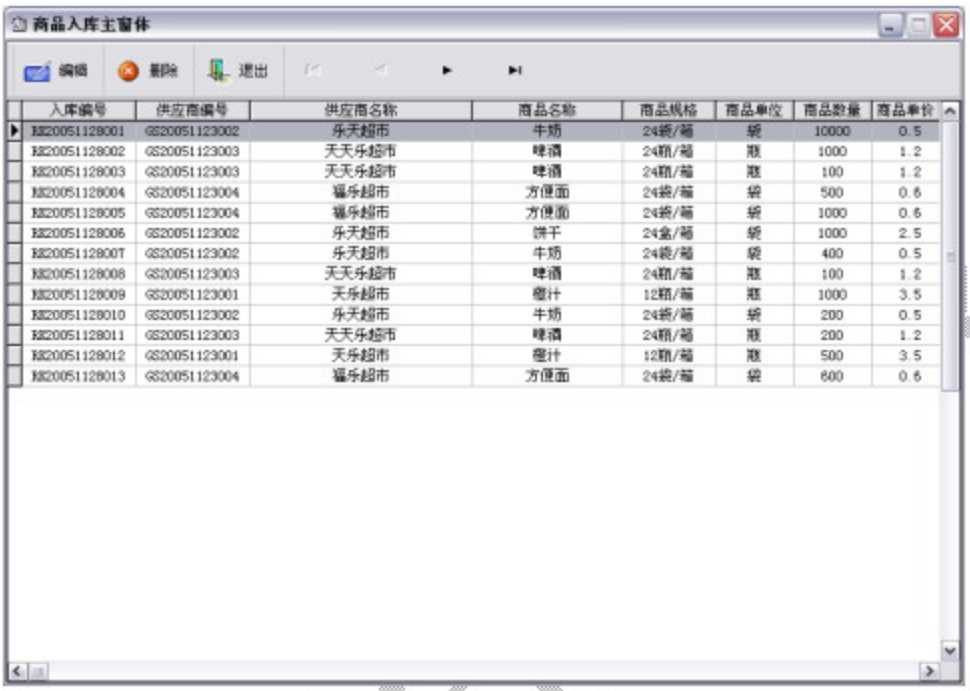


图 17.9　商品入库主窗体运行结果



图 17.10　商品入库编辑窗体运行结果

2. 开发步骤

❑ 商品入库信息主窗体设计

（1）新建一个窗体，在窗体上放置 TToolBar、TDBNavigator、TSpeedButton、TDBGrid

等组件。

（2）设置各主要组件属性如表 17.11 所示。

表 17.11　　组件属性设置

对 象 名	对 象 类 型	属　性	值
SPRKXX_Z	TForm	Font.Color	clBlack
		Font.Size	9
		BorderStyle	bsDialog
DBNavigator1	TDBNavigator	DataSource	DM.DataSource3
SpeedButton1	TSpeedButton	Flat	True
DBGrid1	TDBGrid	DataSource	DataSource3
		Ctl3D	False

❑ 商品入库信息编辑窗体

（1）新建一个窗体，在窗体上放置 TGroupBox、TComboBox、TEdit、TDateTimePicker、TSpeedButton、TLabel 等组件。

（2）设置各主要组件属性如表 17.12 所示。

表 17.12　　组件属性设置

对 象 名	对 象 类 型	属　性	值
SPRKXX	TForm	Font.Color	clBlack
		Font.Size	9
		BorderStyle	bsDialog
SpeedButton1	TSpeedButton	Flat	True

3. 程序相关代码

在窗体创建时打开数据库。

```
procedure TSPRKXX_Z.FormCreate(Sender: TObject);
begin
  with DM.QSPRKXX do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 商品入库信息');
    Open;
  end;
  DM.DSOSPRKXX.Open;
end;
```

单击“编辑”按钮，调出商品信息编辑窗体。

```
procedure TSPRKXX_Z.SpeedButton1Click(Sender: TObject);
begin
  Application.CreateForm(TSPRKXX,SPRKXX);        //创建商品入库信息主窗体
  ActionCD:='M';
  SPRKXX.GetAll;
  SPRKXX.ShowModal;
  SPRKXX.Free;
```

```
end;
```

双击 DBGrid，调出表格相对应的商品信息。

```
procedure TSPRKXX_Z.DBGrid1DblClick(Sender: TObject);
begin
  Application.CreateForm(TSPRKXX,SPRKXX);
  ActionCD:='M';
  SPRKXX.GetAll;
  SPRKXX.ShowModal;
  SPRKXX.Free;
end;
```

下面是单击“添加”按钮的事件代码，用户单击“添加”按钮，入库编号自动生成，并初始化窗体内控件。

```
procedure TSPRKXX.SpeedButton1Click(Sender: TObject);
begin
  GroupBox1.Enabled:=True;                    //控件初始化
  GroupBox2.Enabled:=True;
  SpeedButton3.Enabled:=True;
  SpeedButton4.Enabled:=True;
  ComboBox1.Clear;
  Edit1.Clear;
  Edit2.Clear;
  ComboBox2.Clear;
  Edit3.Clear;
  Edit4.Clear;
  Edit5.Clear;
  Edit6.Clear;
  Edit7.Clear;
  Memo1.Clear;
  ActionCD:= 'N';
  bh;   //自动编号
  ComboBox1.SetFocus;
  SpeedButton1.Enabled:=False;
end;
```

该函数用于入库编号的自动生成。

```
procedure TSPRKXX.BH;          //自动编号
var
  s,m:string;
  i:integer;
begin
  s:= 'RK'+ FormatDateTime('yyyymmdd',Now());
  With DM.QSPRKXX do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select max(入库编号) as ss From 商品入库信息 ');
```

```
    Open;
  end;
  If DM.DSOSPRKXX.FieldByName('入库编号').Value = null then
    s := s + '001'
  else
  begin
    DM.DSOSPRKXX.Last;
    m:= Trim(DM.DSOSPRKXX.FieldByName('入库编号').Value) ;
    i:= StrToInt(Trim(Copy(m,11,8))) ;
    if i<9 then
      s:= s + '00'+ InttoStr(i +1)
    else if i<99 then
      s:= s + '0'+ InttoStr(i +1)
    else
      s:= s +InttoStr(i+1);
  end;
  Edit2.Text := s;
end;
```

下面是单击“保存”按钮的事件代码，用户单击“保存”按钮，会将列表中的数据保存到数据库中。

```
procedure TSPRKXX.SpeedButton3Click(Sender: TObject);
begin
  if (cansave) then
  begin
    Posting; //   提交存盘
    GroupBox1.Enabled:=False;
    GroupBox2.Enabled:=False;
    DM.DSOSPRKXX.Close;
    DM.DSOSPRKXX.Open;
    with DM.QSPRKXX do              //刷新商品入库信息表
    begin
      Close;
      SQL.Clear;
      SQL.Add('Select * from  商品入库信息');
      Open;
    end;
    Application.MessageBox('保存成功!','提示!',64);
    actioncd:='';
    SPRKXX.Close;
  end;
end;
```

该函数用于判断条件是否可以保存，基本思路是判断编辑框或其他字段是否为空，若为空，提示不能为空。

```
function TSPRKXX.CanSave: Boolean;//判断是否保存
begin
```

```
if ComboBox1.Text='' then
begin
  Result:=False;
  Application.MessageBox('供应商编号不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
end;
if Edit1.Text='' then
begin
  Result:=False;
  Application.MessageBox('供应商名称不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
end;
if Edit2.Text='' then
begin
  Result:=False;
  Application.MessageBox('入库编号不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
end;
if ComboBox2.Text='' then
begin
  Result:=False;
  Application.MessageBox('商品名称不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
end;
if Edit3.Text='' then
begin
  Result:=False;
  Application.MessageBox('商品规格不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
end;
if Edit4.Text='' then
begin
  Result:=False;
  Application.MessageBox('商品单位不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
end;
if Edit5.Text='' then
begin
  Result:=False;
  Application.MessageBox('商品单价不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
end;
if Edit6.Text='' then
begin
  Result:=False;
  Application.MessageBox('商品数量不能为空！','提示信息',0+MB_ICONINFORMATION);
  exit;
```

```
    end;
    if Edit7.Text='' then
    begin
      Result:=False;
      Application.MessageBox('经手人不能为空！','提示信息',0+MB_ICONINFORMATION);
      exit;
    end;
  end;
```

该过程是用于向数据库提交数据，并根据库存内的商品名称进行判断。若商品名称相同，则在同商品的库存数量上加上入库的商品数量，若商品名称不同，将入库商品加入库存表。

```
procedure TSPRKXX.Posting;              //提交数据
begin
  if ActionCD='N' then
  begin
    with DM.QSPRKXX do   //入库
    begin
      Close;
      SQL.Clear;
      SQL.Add('INSERT INTO 商品入库信息(供应商编号,供应商名称,商品名称,商品规格,商品单
                           位,商品单价,商品数量,入库日期,经手人,备注,入库编号)');
      SQL.Add('Values(:供应商编号,:供应商名称,:商品名称,:商品规格,:商品单位,:商品单价,:商品数
                           量,:入库日期,:经手人,:备注,:入库编号)');
      Parameters.Items[0].Value:=ComboBox1.Text;
      Parameters.Items[1].Value:=Edit1.Text;
      Parameters.Items[2].Value:=ComboBox2.Text;
      Parameters.Items[3].Value:=Edit3.Text;
      Parameters.Items[4].Value:=Edit4.Text;
      Parameters.Items[5].Value:=StrtoFloat(Edit5.Text);
      Parameters.Items[6].Value:=strtoint(Edit6.Text);
      DateTimePicker1.Date:=date();
      Parameters.Items[7].Value:=FormatDateTime('yyyy-mm-dd',DateTimePicker1.Date);
      Parameters.Items[8].Value:=Edit7.Text;
      Parameters.Items[9].Value:=Memo1.Text;
      Parameters.Items[10].Value:=Edit2.Text;
      Prepared:=True;
      ExecSQL;
      begin                        //获取商品名称到库存表中
        Close;
        SQL.Clear;
        SQL.Add('Select * from 库存表 where 商品名称='+''''+ComboBox2.Text+'''');
        Open;
      end;
      if RecordCount=1 then      //过滤相同数据，若数据有相同的商品名称，将数量相加
      begin
        Edit;
        fieldbyname('商品数量').AsInteger:= fieldbyname('商品数量').AsInteger+strtoint(Edit6.Text);
```

```
            updaterecord;
            post;
            end
          else
          begin                    //将数据插入库存表中
            Close;
            SQL.Clear;
            SQL.Add('INSERT INTO 库存表(商品名称,商品规格,商品单位,商品单价,商品数量)');
            SQL.Add('Values(:商品名称,:商品规格,:商品单位,:商品单价,:商品数量)');
            Parameters.Items[0].Value:=ComboBox2.Text;
            Parameters.Items[1].Value:=Edit3.Text;
            Parameters.Items[2].Value:=Edit4.Text;
            Parameters.Items[3].Value:=strtofloat(Edit5.Text);
            Parameters.Items[4].Value:=strtoint(Edit6.Text);
            Prepared:=True;
            ExecSQL;
          end;
        end;
    end;
end;
```

17.9.2 商品出库管理设计

1. 概述

商品出库管理主要完成用户的提货任务。用户从仓库中提出商品后，库存数量减少。因此，商品出库管理不仅需要记录用户提货的数量、种类等信息，还需要修改库存商品数量。商品出库管理运行结果如图 17.11、17.12 所示。

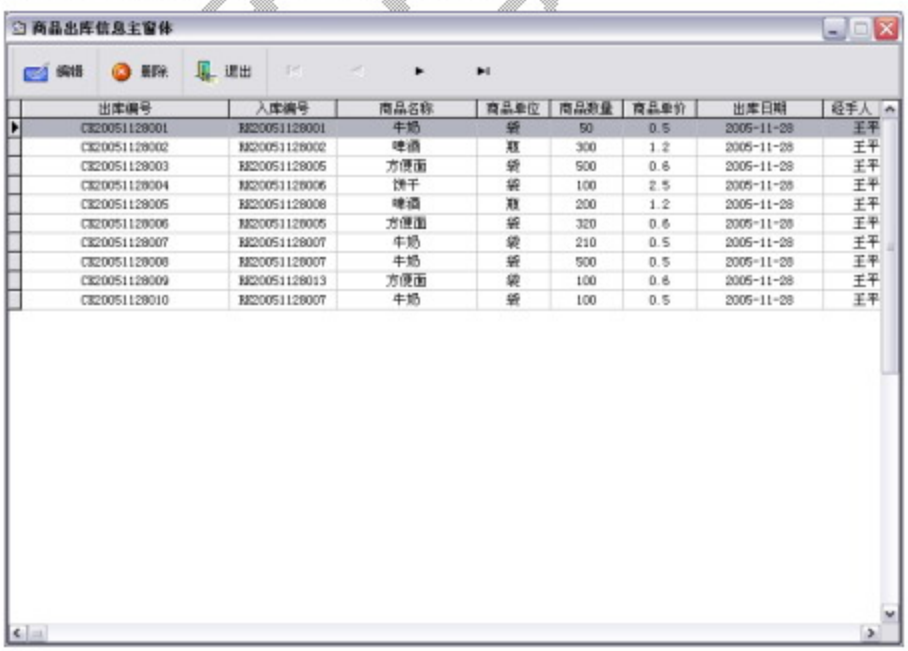


图 17.11 商品出库信息主窗体运行结果



图 17.12 商品出库信息编辑窗体运行结果

2. 设计步骤

❑ 商品出库信息主窗体设计

（1）新建一个窗体，在窗体上放置 TToolBar、TDBNavigator、TSpeedButton、TDBGrid 等组件。

（2）设置各主要组件属性如表 17.13 所示。

表 17.13 组件属性设置

对 象 名	对 象 类 型	属 性	值
SPCKXX_Z	TForm	Font.Color	clBlack
		Font.Size	9
		BorderStyle	bsDialog
DBNavigator1	TDBNavigator	DataSource	DM.DataSource4
SpeedButton1	TSpeedButton	Flat	True
DBGrid1	TDBGrid	DataSource	DataSource4
		Ctl3D	False

❑ 商品出库信息编辑窗体

（1）新建一个窗体，在窗体上放置 TGroupBox、TComboBox、TEdit、TDateTimePicker、TSpeedButton、TLabel 等组件。

（2）设置各主要组件属性如表 17.14 所示。

表 17.14 组件属性设置

对 象 名	对 象 类 型	属 性	值
SPCKXX	TForm	Font.Color	clBlack
		Font.Size	9
		BorderStyle	bsDialog
SpeedButton1	TSpeedButton	Flat	True

3. 程序相关代码

在 Edit5 的退出事件里判断出库数量是否大于库存数量。如果出库数量大于库存数量，

退出操作，否则进行出库操作，库存数量减少。

```
procedure TSPCKXX.Edit5Exit(Sender: TObject);
begin          //判断出库数量与库存数据，出库数量大于库存数量，退出。否则进行数据计算
  with DM.QKCB do
  begin
    Close;
    SQL.Clear;
    SQL.Add('Select 商品数量,商品单价,商品规格,商品单位 from 库存表 where 商品名称
            ='+""+Edit2.Text+"");
    Open;
  end;
  if DM.QKCB.FieldByName('商品数量').Value<strtoint(Edit5.Text) then
  begin
    Application.MessageBox('出库数量不能大于库存数量！','提示！',64);
    Edit5.Clear;
    Edit5.SetFocus;
    exit;
  end
  else
  begin
    DM.QKCB.Edit;
    DM.QKCB.FieldByName('商品数量').Value:=DM.QKCB.FieldByName('商品数量').Value-strtoint(Edit5.Text);
    DM.QKCB.UpdateRecord;
    DM.QKCB.Post;
  end;
end;
```

17.9.3 商品库存查询设计

1. 概述

商品库存查询根据商品名称查询商品库存信息。仓库管理人员通过商品库存查询能够了解库存数量及商品的详细情况。商品库存查询运行结果如图 17.13 所示。



图 17.13 商品库存查询运行结果

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TComboBox、TSpeedButton、TLabel、TBevel、TDBGrid 等组件。

（2）设置各主要组件属性如表 17.15 所示。

表 17.15　　组件属性设置

对 象 名	对 象 类 型	属　性	值
SPKCCX	TForm	Font.Color	clBlack
		Font.Size	9
		BorderStyle	bsDialog
SpeedButton1	TSpeedButton	Flat	True
DBGrid1	TDBGrid	Ctl3D	False

3. 程序相关代码

处理 ComboBox1 的 OnDropDown 事件，当 ComboBox1 处于下拉状态时，将商品名称显示在 ComboBox1 组件中。

```
procedure TSPKCCX.ComboBox1DropDown(Sender: TObject);
var
  i:integer;                              //将商品名称显示在ComboBox中
begin
  ComboBox1.Clear;
  with DM.QKCB do
  begin
    Close;
    SQL.clear;
    SQL.Add('select 商品名称 from 库存表');
    Open;
    if (recordcount > 0) then
      first;
    for i := 0 to recordcount - 1 do
    begin
      combobox1.Items.add(FieldByName('商品名称').asstring);
      next;
    end;
  end;
end;
```

17.10　疑难问题分析解决

17.10.1　使用 TADODataSet 组件时应该注意的问题

当使用 ADODataSet 组件绑定数据库表的字段时，若是在原数据表中修改字段的属性、类型等信息时，要在 ADODataSet 中重新更新其 Fields 属性。操作方法如下。

双击 ADODataSet 组件，弹出如图 17.14 所示的界面，在 ADODataSet 组件编辑器中，单击鼠标右键弹出操作选项，如图 17.15 所示。选择“Add fields”，如图 17.16 所示。单击“OK”按钮，返回如图 17.17 所示界面，完成设置。



图 17.14



图 17.15



图 17.16



图 17.17

17.10.2 双击 DBGrid 表格，如何显示当条数据详细信息

在开发应用程序时，各项信息提供了一个主界面，在主界面中可以对数据进行删除操作，也可以选中某条数据，单击“编辑”按钮，弹出相对应的详细信息。但是这样操作过于麻烦。下面给出了单击 DBGrid 表格，调出当条数据的详细信息代码：

```
procedure TSPJBXX_Z.DBGrid1DblClick(Sender: TObject);
begin
  Application.CreateForm(TSPJBXX,SPJBXX);
  SPJBXX.GetAll;
  ActionCD:='M';
  SPJBXX.ShowModal;
  SPJBXX.Free;
end;
```

但是，怎么才能在相对应的输入框中显示该条数据的信息呢？本实例调用一个自定义函数，代码如下：

```
procedure TSPJBXX.GetAll;//获得所有数据
```

```
begin
    Edit1.Text:=DM.DSOSPXX.FieldByName('商品编号').AsString;
    Edit2.Text:=DM.DSOSPXX.FieldByName('商品名称').AsString;
    Edit3.Text:=DM.DSOSPXX.FieldByName('规格').AsString;
    Edit4.Text:=DM.DSOSPXX.FieldByName('保质期').AsString;
    Edit5.Text:=DM.DSOSPXX.fieldByName('产地').AsString;
    Edit6.Text:=DM.DSOSPXX.FieldByName('单位').AsString;
    Edit7.Text:=DM.DSOSPXX.FieldByName('商品单价').AsString;
    Edit8.Text:=DM.DSOSPXX.FieldByName('批发价').AsString;
    Edit9.Text:=DM.DSOSPXX.FieldByName('零售价').AsString;
    Edit10.Text:=DM.DSOSPXX.FieldByName('商品简称').AsString;
    Edit11.Text:=DM.DSOSPXX.FieldByName('供应商名称').AsString;
    ComboBox1.Text:=DM.DSOSPXX.FieldByName('供应商编号').AsString;
    Memo1.Text:=DM.DSOSPXX.FieldByName('备注').AsString;
end;
```

这样，双击 DBGrid 表中的任何一条数据，都会弹出相对应数据的详细信息了。

17.10.3 如何判断数据库中是否有相同字段的数据

在商品入库录入中，要把商品数量同时存入“商品入库信息表”和“库存表”中。若想在“库存表”方便地查看同一商品的信息，就要根据商品名称来判断库存表中是否有相同的商品，若没有相同的商品，进行插入操作，代码如下：

```
begin                    //获取商品名称到库存表中
        Close;
        SQL.Clear;
        SQL.Add('Select * from 库存表 where 商品名称='+''''+ComboBox2.Text+'''');
        Open;
      end;
    if RecordCount=1 then      //过滤相同数据，若有相同的商品名称，将数量相加
    begin
      Edit;
      fieldbyname('商品数量').AsInteger:= fieldbyname('商品数量').AsInteger+strtoint(Edit6.Text);
      updaterecord;
      post;
      end
    else
    begin                  //将数据插入库存表中
      Close;
      SQL.Clear;
      SQL.Add('INSERT INTO 库存表(商品名称,商品规格,商品单位,商品单价,商品数量)');
      SQL.Add('Values(:商品名称,:商品规格,:商品单位,:商品单价,:商品数量)');
      Parameters.Items[0].Value:=ComboBox2.Text;
      Parameters.Items[1].Value:=Edit3.Text;
      Parameters.Items[2].Value:=Edit4.Text;
      Parameters.Items[3].Value:=strtofloat(Edit5.Text);
      Parameters.Items[4].Value:=strtoint(Edit6.Text);
      Prepared:=True;
```

```
        ExecSQL;
      end;
```

17.10.4 判断进价、批发价、零售价是否输入合法

用户在商品基本信息中存在“商品进价”、“商品批发价”以及“商品零售价”。因价格不同，所以必须要判断输入的价格是否合理，下面列出部分代码：

```
function TSPJBXX.CanSave: Boolean;//判断是否保存
begin


  if StrToFloat(Edit7.Text)>StrToFloat(Edit8.Text) then
  begin
    Result:=False;
    Application.MessageBox('进价不能大于批发价！','提示信息',0+MB_ICONINFORMATION);
    exit;
  end;
  if StrToFloat(Edit8.Text)>StrToFloat(Edit9.Text) then
  begin
    Result:=False;
    Application.MessageBox('批发价不能大于零售价！','提示信息',0+MB_ICONINFORMATION);
    exit;
  end;
  if StrToFloat(Edit7.Text)>StrToFloat(Edit9.Text) then
  begin
    Result:=False;
    Application.MessageBox('进价不能大于零售价！','提示信息',0+MB_ICONINFORMATION);
    exit;
  end;
end;
```

17.11 程序测试常见错误解决

在开发完一个应用程序后，开发人员都希望它能够正常运行，但是在测试程序的时候，程序运行时总会出现这样或那样的错误。下面归纳出两条在编程中容易出现的错误。

- 赋值语句类型不兼容。
- 进行数据备份或数据恢复时，SQL 语句中要注意的问题。

下面将以具体实例说明在程序调试中的问题。

以下是商品基本信息中“保存”按钮的调用函数的部分原始代码：

```
function TSPJBXX.CanSave: Boolean;//判断是否保存
begin


  if Edit7.Text>Edit8.Text then
```

```
    begin
      Result:=False;
      Application.MessageBox('进价不能大于批发价！','提示信息',0+MB_ICONINFORMATION);
      exit;
    end;
    if Edit8.Text>Edit9.Text then
    begin
      Result:=False;
      Application.MessageBox('批发价不能大于零售价！','提示信息',0+MB_ICONINFORMATION);
      exit;
    end;
    if Edit7.Text>Edit9.Text then
    begin
      Result:=False;
      Application.MessageBox('进价不能大于零售价！','提示信息',0+MB_ICONINFORMATION);
      exit;
    end;

end;
```

在商品基本信息编辑窗体中，输入“进价”、“批发价”、“零售价”，如图 17.18 所示。



图 17.18　录入信息

单击“保存”按钮，提示错误如图 17.19 所示：



图 17.19　提示错误

最后修改程序代码如下：

```
function TSPJBXX.CanSave: Boolean;//判断是否保存
begin
  if StrToFloat(Edit7.Text)>StrToFloat(Edit8.Text) then
  begin
    Result:=False;
    Application.MessageBox('进价不能大于批发价！','提示信息',0+MB_ICONINFORMATION);
    exit;
  end;
  if StrToFloat(Edit8.Text)>StrToFloat(Edit9.Text) then
  begin
    Result:=False;
    Application.MessageBox('批发价不能大于零售价！','提示信息',0+MB_ICONINFORMATION);
    exit;
  end;
  if StrToFloat(Edit7.Text)>StrToFloat(Edit9.Text) then
  begin
    Result:=False;
    Application.MessageBox('进价不能大于零售价！','提示信息',0+MB_ICONINFORMATION);
    exit;
  end;
end;
```

在数据备份或数据恢复时，要针对程序中所用的数据库进行数据备份或恢复操作。由于开发者对数据库命名都有自己的习惯性，有的开发者比较习惯用汉字命名数据库名，有的开发者习惯用英文或拼音命名数据库名。这样一来，在数据备份或数据恢复中的 SQL 语句中，就会有很大的不同之处，下面给出数据备份的源代码：

```
procedure TMAIN.SpeedButton7Click(Sender: TObject);
var
  Str_Path:string;
begin
  SaveDialog1.Filter := '所有文件(*.*)|*.*';
  SaveDialog1.Title := '选择备份路径和文件名';
  if SaveDialog1.Execute=true then
    begin
      Str_Path:=SaveDialog1.FileName;
    end;
  if Str_Path<>'' then
    begin
      with DM.ADOQuery1 do
        begin
          try
            Screen.Cursor := crHourGlass;
            DM.ADOQuery1.Close;
            DM.ADOQuery1.SQL.Clear;
            DM.ADOQuery1.SQL.Add(' BACKUP DATABASE  '+CKGL'+'  TO DISK='''+Str_Path+'''');
```

```
                DM.ADOQuery1.ExecSQL();
                Screen.Cursor := crDefault;
                Application.MessageBox('数据库备份成功完成','数据库备份',0+mb_iconinformation);
            except
                Screen.Cursor := crDefault;
                Application.MessageBox('数据库备份失败！请检查备份路径或网络状态','数据库备份
                                    ',0+mb_iconinformation);
                exit;
            end;
        end;
    end;
end;
```

值得注意的是这行代码：

```
DM.ADOQuery1.SQL.Add(' BACKUP DATABASE   '+'CKGL'+'   TO DISK ='" +Str_Path+"');
```

其中用“"”引起的为数据库名。若是用汉字命名的数据库，这样的写法是正确的，但是以英文命名的数据库，这样写法就大错特错了。最后修改程序代码如下：

```
DM.ADOQuery1.SQL.Add(' BACKUP DATABASE   CKGL   TO DISK ='+'" +Str_Path+"');
```

17.12 程序设计清单

仓库管理系统文件清单如表 17.16 所示。

表 17.16　　仓库管理设计清单

文 件 名	文 件 类 型	说　明
MAIN.dfm	窗体文件	主窗体
UABOUT.dfm	窗体文件	关于窗体
UGYSXX.dfm	窗体文件	供应商基本信息编辑窗体
UGYSXX_Z.dfm	窗体文件	供应商基本信息主窗体
USPJBXX.dfm	窗体文件	商品基本信息编辑窗体
USPJBXX_Z.dfm	窗体文件	商品基本信息主窗体
USPCKXX.dfm	窗体文件	商品出库信息编辑窗体
USPCKXX_Z.dfm	窗体文件	商品出库信息主窗体
USPRKXX.dfm	窗体文件	商品入库信息编辑窗体
USPRKXX_Z.dfm	窗体文件	商品入库信息主窗体
USPKCCX.dfm	窗体文件	商品库存查询窗体
UMMXG.dfm	窗体文件	密码修改设置窗体
UDM.dfm	窗体文件	数据模块窗体
USC.dfm	窗体文件	启动窗体
ULOGIN.dfm	窗体文件	登录窗体
CKGL.dpr	工程文件	
CKGL.exe	可执行文件	

17.13 技术、经验总结

17.13.1 技术总结

在 TEdit 中防止输入非法字符。在 TEdit 的 OnKeyPress 事件中添加如下代码：

```
procedure TSPJBXX.Edit7KeyPress(Sender: TObject; var Key: Char);
begin
  if not (key in['0'..'9','.',#13,#8]) then
    key:=#0;
end;
```

17.13.2 经验总结

使用 TDBGrid 组件显示数据时，应该设置 TDBGrid 的“Options.dgEding”属性为“False”，以防止用户不小心修改数据。设置 TDBGrid 的“Options.dgRowSelect”属性为“True”，可以单击整行数据。

明日科技

- 能够进行商品盘点
- 实现商品查询管理
- 能够进行结款管理
- 实现报表打印

18.2 系统分析

18.2.1 系统规划

1. 系统目标

利用计算机实现商品销售，取代人工管理，避免计算失误。

2. 系统功能

- 基础信息管理

在基础信息管理中包含商品信息管理、客户信息管理、操作员管理和库存管理。在商品信息管理中需要实现商品信息的添加、浏览、删除。在库存管理中需要实现商品入库管理。

- 销售管理

在销售管理中需要实现客户订货、商品销售、销售退货、客户结款等功能。客户订货时，可以一次订购多种商品。

- 查询管理

在查询管理中包括商品销售查询、销售退货查询、客户结款查询、超期应收款查询。用户在查询信息时，可以选择多种条件查询。

- 报表管理

在报表管理中需要实现商品日销售报表、月销售报表、单商品销售报表。

3. 系统结构图

销售管理系统结构图如图 18.1 所示。

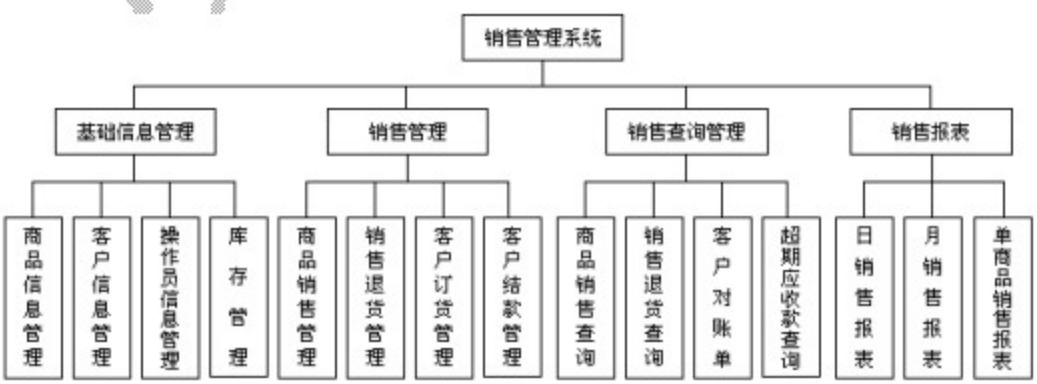


图 18.1 销售管理系统结构图

18.2.2 系统流程图

销售管理系统流程图如图 18.2 所示。

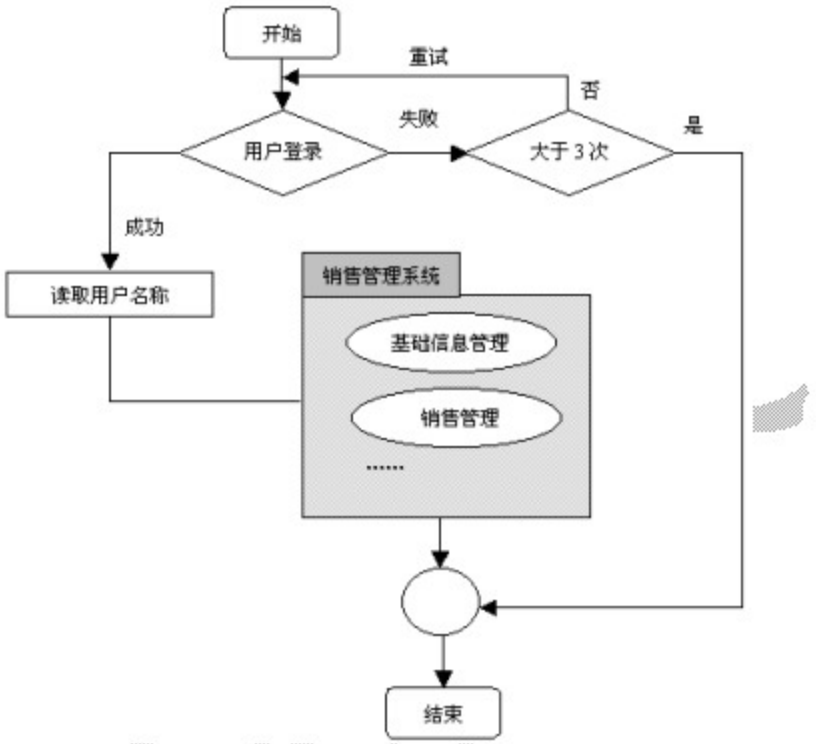


图 18.2 销售管理系统流程图

18.2.3 业务流程

销售管理系统业务流程如图 18.3 所示。

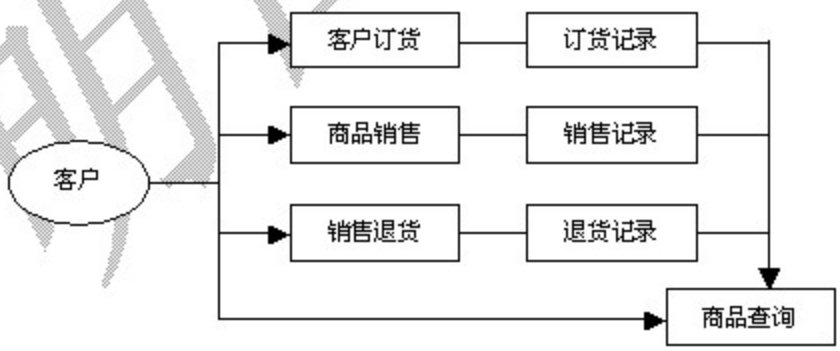


图 18.3 销售管理系统业务流程图

18.3 数据库设计

18.3.1 数据库概念设计

销售管理系统的 E-R 图如图 18.4 所示。



图 18.4　销售管理系统 E-R 图

18.3.2　数据库逻辑结构设计

销售管理系统共使用了 12 张数据表，分别为：销售退货表、销售退单票号表、客户基础信息表、订货票号表、销售票号表、库存表、销售表、商品基础信息表、客户结款表、客户订货表、客户结款记录表、操作员信息表。由于篇幅关系，在此只列出使用较多的数据表结构（如表 18.1～表 18.4 所示），其他表结构请参考光盘。

表 18.1　　销售退单票号表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
销售退单票号	Varchar(30)	是		否	
品种数	Int				
数量	Int				
应付金额	Money				
付款方式	Varchar(20)				
实付金额	Money				
退货时间	DateTime				GetDate()
经手人	Varchar(20)				
操作员	Varchar(20)				

表 18.2　　销售退货表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
商品编号	Varchar(30)				
商品名称	Varchar(30)				
客户编号	Varchar(30)				
客户全称	Varchar(30)				
数量	Int				
金额	Money				
折扣	Real				
销售退单票号	Varchar(30)		是		

表 18.3　　销售票号表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
销售票号	Varchar(30)	是		否	
品种数	Int				
数量	Int				

续表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
应付金额	Money				
付款方式	Varchar(20)				
实付金额	Money				
销售时间	DateTime				GetDate()
经手人	Varchar(20)				
操作员	Varchar(20)				

表 18.4　　销售表

字段名称	字段类型	主　键	外　键	是否为空	默　认　值
商品编号	Varchar(30)				
商品名称	Varchar(30)				
客户编号	Varchar(30)				
客户全称	Varchar(30)				
数量	Int				
金额	Money				
折扣	Real				
销售票号	Varchar(30)		是		

18.4　主窗体设计

主窗体设计步骤如下。

（1）创建一个窗体，在该窗体上放置 TMainMenu、TSplitter、TImageList、TlistView、TCoolBar、TToolBar、TStatusBar、TSpeedButton 等组件，如图 18.5 所示。

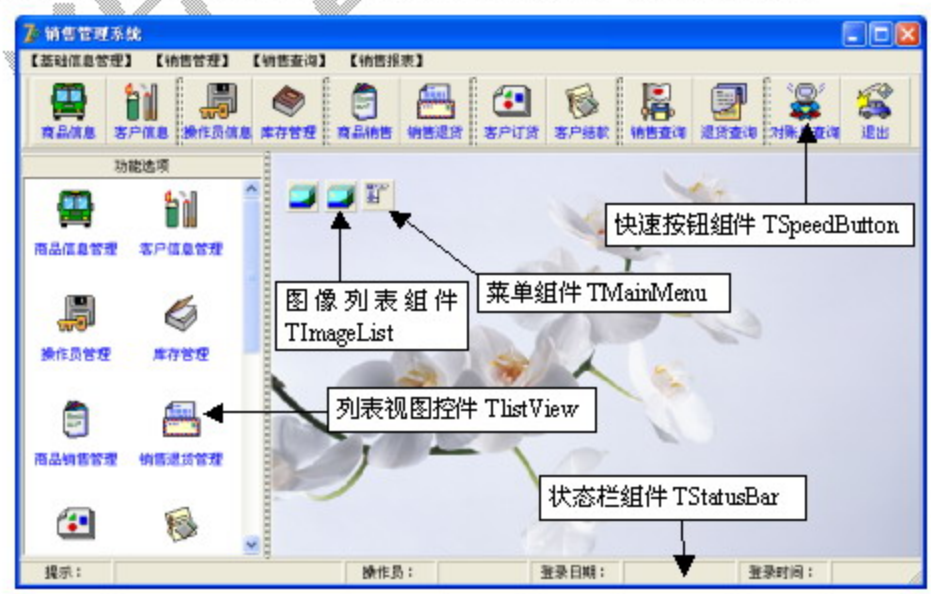


图 18.5　主程序界面

（2）设置各主要组件属性如表 18.5 所示。

表 18.5　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form89	TForm	Font.Color	clBlue
		Font.Size	8
ListView1	TListView	LargeImages	ImageList1
MainMenu1	TMainMenu	Images	ImageList2

18.5　状态栏设计

状态栏设计步骤如下。

（1）在窗体中选中 StatusBar1 组件，双击该组件，打开面板编辑器，效果如图 18.6 所示。

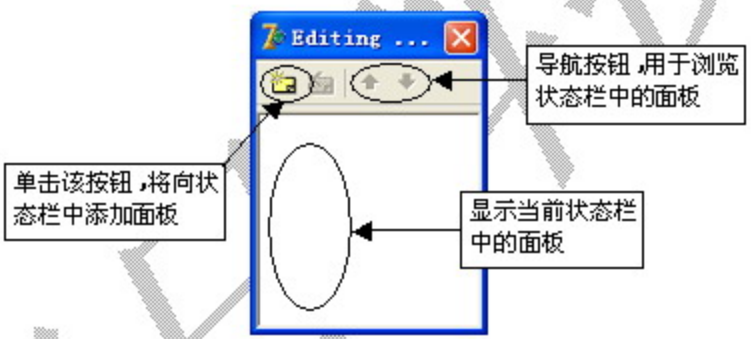


图 18.6　面板编辑器

（2）单击“New Add”按钮，向状态栏中添加面板，在面板编辑器中选中新添加的面板，在对象观察器中设置“Text”属性，例如，“提示：”，修改“AlignMent”属性为“taCenter”，使面板内容居中显示。效果如图 18.7、图 18.8 所示。



图 18.7　对象观察器



图 18.8　面板编辑窗口

（3）按照上面的方法创建其他面板，效果如图 18.5 所示。

18.6 数据模块设计

数据模块设计步骤如下。

（1）启动 Dephi 7，打开工程“xs”，单击“File”/“New”/“Data Module”菜单项，创建数据模块。

（2）选中数据模块，在对象观察器中修改“Name”属性为“Data”。

（3）在数据模块上放置 TADOConnection 组件、TADOQuery 组件，如图 18.9 所示。

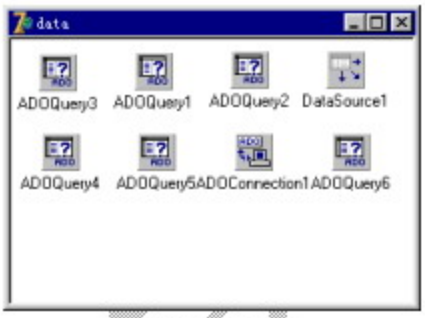


图 18.9 数据模块

（4）设置各主要组件属性如表 18.6 所示。

表 18.6 组件属性设置

对 象 名	对象类型	属 性	值
ADOConnection1	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=销售管理系统
ADOQuery1	TADOQuery	Connection	ADOConnection1
ADOQuery2	TADOQuery	Connection	ADOConnection1
ADOQuery3	TADOQuery	Connection	ADOConnection1

（5）单击“Save All”按钮，将数据单元保存为“DataModal”。

18.7 销售报表设计

销售报表设计步骤如下。

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TSpeedButton、TComboBox、TCheckBox、TDateTimePicker、TRVSystem 等组件，如图 18.10 所示。

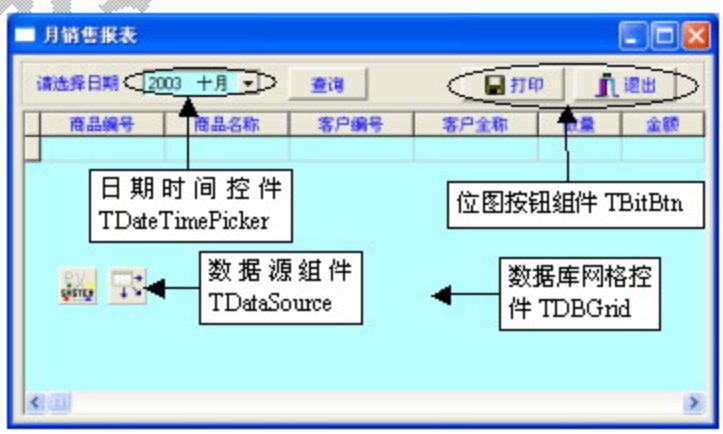


图 18.10 月销售报表

（2）设置各主要组件属性如表 18.7 所示。

表 18.7　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form62	TForm	Font.Color	clBlue
		Font.Size	10
		BorderStyle	bsDialog
BitBtn8	TBitBtn	Kind	bkClose
DBGrid1	TDBGrid	DataSource	DataSource1

（3）程序相关代码

下面是 TRVSystem 组件的 OnPrint 事件代码，当打印时，打印机会根据下面的设置进行打印。

```
procedure TForm62.aaPrint(Sender: TObject);
begin
  if Data.ADOQuery1.Active = True then
  begin
    with Sender as TBaseRePort do
    begin
      setfont('宋体',18);//设置字体大小
      TBaseRePort(sender).Canvas.Font.Color := clRed;//设置字体颜色
      Bold := True;//粗体
      underline := True;//下划线
      ypos := 1.0;//标题上边距
      PrintCenter('月销售报表',pagewidth / 2);//居中打印报表标题
      setFont('宋体',10);
      TBaseRePort(sender).Canvas.Font.Color := clBlue;
      ypos := 1.4 ;
      printLeft('操作员姓名:',0.9);
      printleft(czy,1.9);
      printleft('时间:',pageWidth-2.2);
      PrintLeft(FormatDateTime('yyyy-mm-dd',Now),pageWidth-1.7);
      ypos := 1.7;
      Cleartabs;
      setFont('宋体',12);
      TBaseRePort(sender).Canvas.Font.Color := clBlue;
      setpen(clblack,pssolid,1,pmcopy);//设置画笔风格
      settab(0.9,pjcenter,1.1,0,boxlineall,0);//设置制表位
      settab(na,pjcenter,1.0,0,boxlineall,0);
      settab(na,pjcenter,1.0,0,boxlineall,0);
      settab(na,pjcenter,1.0,0,boxlineall,0);
      settab(na,pjcenter,0.6,0,boxlineall,0);
      settab(na,pjcenter,0.6,0,boxlineall,0);
      settab(na,pjcenter,1.2,0,boxlineall,0);
      bold := True;
      tab(-2,na,-2,-2,10);//定位制表位
      Print('商品编号');
```

```
        tab(na,na,-2,-2,10);
        Print('商品名称');
        Tab(na,na,-2,-2,10);
        Print('客户编号');
        Tab(na,na,-2,-2,10);
        Print('客户全称');
        Tab(na,na,-2,-2,10);
        Print('数量');
        Tab(na,na,-2,-2,10);
        Print('金额');
        Tab(na,-2,-2,-2,10);
        Print('销售票号');
        Bold := False;
        ypos := 1.9;
        ClearTabs;
        setFont('宋体',10);
        TBaseRePort(sender).Canvas.Font.Color := clBlue;
        settab(0.9,pjcenter,1.1,2,boxlineall,0);
        settab(na,pjcenter,1.0,2,boxlineall,0);
        settab(na,pjcenter,1.0,2,boxlineall,0);
        settab(na,pjcenter,1.0,2,boxlineall,0);
        settab(na,pjcenter,0.6,2,boxlineall,0);
        settab(na,pjcenter,0.6,2,boxlineall,0);
        settab(na,pjcenter,1.2,2,boxlineall,0);
        Data.ADOQuery1.First;
        ypos := 1.85;
        while Not Data.ADOQuery1.Eof do
        begin
          Tab(-2,na,na,na,0);
          print(Data.ADOQuery1.Fields[0].Value);
          tab(na,na,na,na,0);
          print(Data.ADOQuery1.Fields[1].Value);
          tab(na,na,na,na,0);
          print(Data.ADOQuery1.Fields[2].Value);
          tab(na,na,na,na,0);
          print(Data.ADOQuery1.Fields[3].Value);
          tab(na,na,na,na,0);
          print(FloatToStr(Data.ADOQuery1.Fields[4].Value));
          tab(na,na,na,na,0);
          print(FloatToStr(Data.ADOQuery1.Fields[5].Value));
          tab(na,-2,na,na,0);
          print(Data.ADOQuery1.Fields[7].Value);
          Println('');//换行打印
          Data.ADOQuery1.Next;
        end;
      end;
    end;
  end;
```

18.8 程序设计与编码

18.8.1 商品销售管理设计

1. 概述

商品销售管理主要实现将库存商品销售给客户。在进行商品销售时，程序会修改相应商品库存，同时记录销售信息（商品名称、销售数量、销售金额以及客户相关信息），此外，在该模块中还可以按订单销售商品。商品销售管理运行结果如图 18.11 所示。

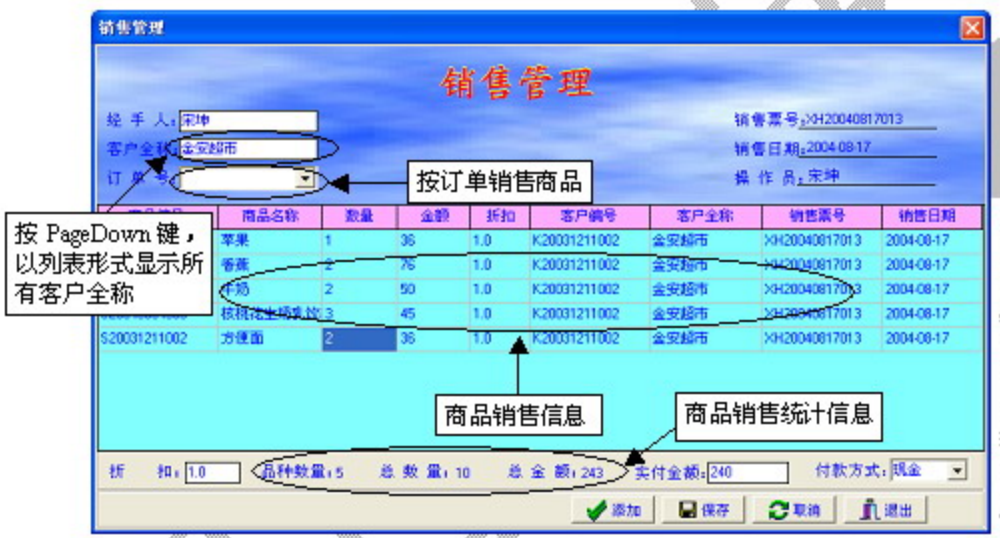


图 18.11 销售管理运行结果

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TStringGrid、TDataSource、TDBGrid、TListBox、TShape 等组件。

（2）设置各主要组件属性，如表 18.8 所示。

表 18.8 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form24	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
ListBox1	TListBox	Visible	False
DBGrid1	TDBGrid	Visible	False
		DataSource	DataSource1
Edit8	TEdit	Text	1.0

3. 程序相关代码

窗体显示时，显示当前用户，向列表框中添加客户全称，向下拉列表框中添加订单号。

在进行商品销售时，需要记录操作员信息，所以在窗体显示时加载当前操作员。为了方便用户录入客户信息，在“客户全称”编辑框中按 PageDown 键将以列表框形式显示客户信息，供用户选择，此处在窗体显示时将客户信息查询出来，将其添加到列表框中，目的是为了实现上述功能。在销售商品时，用户也可以按订货信息销售商品，在讲解销售部分的代码时会进行详细介绍。窗体显示时，向“订单号”组合框中加载订单号，这些订单号表示已完成订货而没有进行销售的订货信息，在查询订单号时通过“是否可用”字段标识定单（为 0 标识还没有进行销售）。

```
procedure TForm24.FormShow(Sender: TObject);
begin
  Label21.Caption:= czy;
  ListBox1.Items.Clear;//清空列表框中的内容
    with Data.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select distinct 客户全称 from 客户基础信息表');
      Open;
    end;
    while not Data.ADOQuery1.Eof do
    begin
      //向列表中添加数据
      ListBox1.Items.Add(Data.ADOQuery1.FieldByName('客户全称').AsString);
      Data.ADOQuery1.Next;
    end;
    ComboBox3.Clear;//清空组合框中的内容
    with Data.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 订货票号表 where 是否可用 = 0');
      Open;
    end;
    if Data.ADOQuery1.RecordCount>0 then
    while not Data.ADOQuery1.Eof do
    begin
      //向组合框中添加数据
      ComboBox3.Items.Add(Data.ADOQuery1.FieldByName('订单号').AsString);
      Data.ADOQuery1.Next;
    end;
  ComboBox2.ItemIndex := 0;
  BitBtn3.SetFocus;
end;
```

下面是“保存”按钮的单击事件代码。相对于其他功能来说，销售过程较为复杂。因为在销售过程中需要区分是直接销售还是按订单销售，程序通过全局变量 isDy 进行标识。在直接销售商品时，不需要输入订单号，而按订单销售时，用户需要在“订单号”组合框

中选择一个订单号，按 Enter 键会调出订单信息。按订单销售时不需要输入“客户全称”，因为在订单信息中已经包含了客户信息，而且也不能在表格中修改订单信息。

无论直接销售还是按订单销售，都存在一些共同的操作，如检查信息是否有误（信息是否为空，销售数量是否大于库存数量）、生成销售票号，修改商品库存，保存销售信息、修改与客户的账目关系，惟一不同的是，在按订单销售时，还需要修改订单标识，将订单号的“是否可用”标识设置为 1，表示已经销售了，不允许再进行销售。

经过对上面的分析，知道需要对多个表同时进行操作，为了保证数据的完整性，使用了事务进行控制，详细代码如下：

```
procedure TForm24.BitBtn4Click(Sender: TObject);
var
  i,Sum: Integer;
begin
  if Trim(Edit4.Text)<>'' then
  begin
    if JCxmlb = True then
    begin
      For i := 1 to StringGrid1.RowCount-1 do //利用循环判断表格中商品的数量是否大于库存数量
      begin
        Sum := GroupSL(StringGrid1.Cells[0,i]);
        with Data.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('select * from 库存表 where 商品编号 = :a');
          Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,i]);
          Open;
        end;
        if Sum >Data.ADOQuery1.FieldByName('库存数量').Value then
        begin
          if Isdy = True then
          begin
            Application.MessageBox(Pchar('商品编号为'+Trim(StringGrid1.Cells[0,i])+'的商
              品数量大于库存数量，不能完成订单销售。'),'提示',64);
            Exit;
          end
          else
          begin
            Application.MessageBox(Pchar('商品编号为'+Trim(StringGrid1.Cells[0,i])+'的商
              品数量大于库存数量，不能完成销售操作。'),'提示',64);
            Exit;
          end;
        end;
      end;
      Data.ADOConnection1.BeginTrans;//开始一个事务
      Try //异常捕捉
```

```
with Data.ADOQuery1 do
begin
  Close;
  SQL.Clear;
  SQL.Add('Insert 销售票号表 Values(:a,:b,:c,:d,:e,:f,Default,:h,:i)');
  Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[7,1]);
  Parameters.ParamByName('b').Value := StrToInt(Label18.Caption);
  Parameters.ParamByName('c').Value := StrToInt(Label18.Caption);
  Parameters.ParamByName('d').Value := StrToFloat(Label19.Caption);
  Parameters.ParamByName('e').Value := Trim(ComboBox2.Text);
  Parameters.ParamByName('f').Value := StrToFloat(Edit2.Text);
  Parameters.ParamByName('h').Value := Trim(Edit4.Text);
  Parameters.ParamByName('i').Value := Trim(Label21.Caption);
  ExecSQL;//执行SQL语句
end;
For i := 1 to StringGrid1.RowCount-1 do
begin
  with Data.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('Insert 销售表 values(:a,:b,:c,:d,:e,:f,:g,:h)');
    Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,1]);
    Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[1,1]);
    Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[5,1]);
    Parameters.ParamByName('d').Value := Trim(StringGrid1.Cells[6,1]);
    Parameters.ParamByName('e').Value := StrToInt(StringGrid1.Cells[2,i]);
    Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[3,i]);
    Parameters.ParamByName('g').Value := StrToFloat(StringGrid1.Cells[4,i]);
    Parameters.ParamByName('h').Value := Trim(StringGrid1.Cells[7,1]);
    ExecSQL;
  end;
end;
For i := 1 to StringGrid1.RowCount-1 do
begin
  with Data.ADOQuery2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 商品基础信息表 where 商品编号 = :a');
    Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,i]);
    Open;
  end;
  jj := Data.ADOQuery2.FieldByName('进价').Value;
  with Data.ADOQuery2 do
  begin
    Close;
```

```
      SQL.Clear;
      SQL.Add('update 库存表 set 库存数量 = 库存数量 - :a,库存金额 =库存金额 -:b
      where 商品编号 = :c ');
      Parameters.ParamByName('a').Value := StrToInt(StringGrid1.Cells[2,i]);
      Parameters.ParamByName('b').Value := jj*StrToInt(StringGrid1.Cells[2,i]);
      Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[0,i]);
      ExecSQL;
    end;
end;
with Data.ADOQuery2 do
begin
   Close;
   SQL.Clear;
   SQL.Add('select * from 客户结款表 where 客户编号 = :a');
   Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[5,1]);
   Open;
end;
if Data.ADOQuery2.RecordCount <1 then
begin
   with Data.ADOQuery1 do
   begin
      Close;
      SQL.Clear;
      SQL.Add('Insert 客户结款表 Values (:a ,:b ,:c ,:d,:e,default)');
      Parameters.ParamByName('a').Value :=   Trim(StringGrid1.Cells[5,1]);
      Parameters.ParamByName('b').Value :=   Trim(StringGrid1.Cells[6,1]);
      Parameters.ParamByName('c').Value :=   StrToFloat(Label19.Caption);
      Parameters.ParamByName('d').Value := StrToFloat(Label19.Caption)-
        StrToFloat(Edit2.Text);
      Parameters.ParamByName('e').Value :=   StrToFloat(Edit2.Text);
      ExecSQL;
   end;
end
else
begin
   with Data.ADOQuery1 do
   begin
      Close;
      SQL.Clear;
      SQL.Add(' update 客户结款表 set 金额小计 = 金额小计 + :a ,欠款记账 = 欠款记账
          + :b ,付款合计 = 付款合计 + :c ,结款时间 = Default where 客户编号 = :d');
      Parameters.ParamByName('a').Value :=   StrToFloat(Label19.Caption);
      Parameters.ParamByName('b').Value :=   StrToFloat(Label19.Caption)-
        StrToFloat(Edit2.Text);
      Parameters.ParamByName('c').Value :=   StrToFloat(Edit2.Text);
      Parameters.ParamByName('d').Value := Trim(StringGrid1.Cells[5,1]);
      ExecSQL;
```

```
            end;
          end;
          if isDy = True then
          begin
            With Data.ADOQuery2 do
            begin
              Close;
              SQL.Clear;
              SQL.Add('update 订货票号表 set 是否可用 = 1 where 订单号 = :a');
              Parameters.ParamByName('a').Value := ddh;
              ExecSQL;
            end;
            ComboBox3.Clear;
          end;
        Data.ADOConnection1.CommitTrans;//提交事务
        ApplicationMessageBox('保存成功。','提示',0+64);
        ComboBox3.Clear;
         with Data.ADOQuery1 do
         begin
            Close;
            SQL.Clear;
            SQL.Add('select * from 订货票号表 where 是否可用 = 0');
            Open;
         end;
         if Data.ADOQuery1.RecordCount>0 then
         while not Data.ADOQuery1.Eof do
         begin
            ComboBox3.Items.Add(Data.ADOQuery1.FieldByName('订单号').AsString);
            Data.ADOQuery1.Next;
         end;
        Clear;
      Except //处理异常
        Data.ADOConnection1.RollbackTrans;//事务回滚
        Application.MessageBox('系统出错。','提示',0+64);
        Close;
      end;
  end
  else
  begin
     Application.MessageBox('项目列表有误。','提示',64);
     Exit;
  end;
end
else
begin
   Application.MessageBox('经手人不能为空。','提示',64);
   Edit4.SetFocus;
```

```
    end;
end;
```

18.8.2 销售退货管理设计

1. 概述

企业在销售商品后，由于质量等方面的原因，总会发生一些退货的情况。销售退货管理主要进行客户退货处理。在进行退货处理时，程序会记录退货商品名称、退货数量、退货金额以及客户相关信息，同时还会修改库存，进行结款。商品退货管理如图 18.12 所示。



图 18.12 销售退货运行结果

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TStringGrid、TDataSource、TDBGrid、TListBox、TShape 等组件。

（2）设置各主要组件属性如表 18.9 所示。

表 18.9 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form27	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
ListBox1	TListBox	Visible	False
DBGrid1	TDBGrid	Visible	False
		DataSource	DataSource1
Edit8	TEdit	Text	1.0

3. 程序相关代码

在窗体显示时加载当前用户，向列表框中添加客户全称。当用户在“客户全称”编辑框中按 PageDown 键将以列表框形式显示客户信息，这些客户信息便是在窗体显示时

加载的。

```
procedure TForm27.FormShow(Sender: TObject);
begin
  Label21.Caption:= czy;
  ListBox1.Items.Clear;
  with Data.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select distinct 客户全称 from 销售表');
    Open;
  end;
  while not Data.ADOQuery1.Eof do
  begin
    ListBox1.Items.Add(Data.ADOQuery1.FieldByName('客户全称').AsString);
    Data.ADOQuery1.Next;
  end;
  ComboBox2.ItemIndex := 0;
  BitBtn3.SetFocus;
end;
```

下面是“保存”按钮的单击事件代码，程序首先判断操作员输入的数据是否有误，如果没有错误，会将退货信息保存到数据库中。在进行商品退货时，需要记录退货信息、修改商品库存，修改与客户的账目关系，整个操作需要修改“销售退单票号表”、“销售退货表”、“库存表”、“客户结款表”等几个数据表，为了保证数据完整性，整个操作采用事务进行控制，在操作过程中，任何一个环节出现错误，都将取消操作（通过回滚事务）。

```
procedure TForm27.BitBtn4Click(Sender: TObject);
var
  row: Integer;
  SPjg: Real;//记录商品进价，用来修改库存金额
begin
  if (Trim(Edit4.Text)='')or (Jcxmlb = False)or(Trim(ComboBox2.Text)='')then
  begin
    Application.MessageBox('项目列表有误。','提示',0+64);
    Exit;
  end;
  Data.ADOConnection1.BeginTrans; //开始一个事务
  try
  with Data.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('Insert 销售退单票号表 values (:a,:b,:c,:d,:e,:f,Default,:g,:h)');
    Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[7,1]);
    Parameters.ParamByName('b').Value := StrToInt(Label18.Caption);
    Parameters.ParamByName('c').Value := StrToInt(Label18.Caption);
```

```
    Parameters.ParamByName('d').Value := StrToFloat(Label19.Caption);
    Parameters.ParamByName('e').Value := ComboBox2.Text;
    Parameters.ParamByName('f').Value := StrToFloat(Edit2.Text);
    Parameters.ParamByName('g').Value := Trim(Edit4.Text);
    Parameters.ParamByName('h').Value := Trim(Label21.Caption);
    ExecSQL;
  end;
  For Row := 1 to StringGrid1.RowCount-1 do
  begin
    with Data.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Insert 销售退货表 values(:a,:b,:c,:d,:e,:f,:g,:h)');
      Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,Row]);
      Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[1,Row]);
      Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[5,Row]);
      Parameters.ParamByName('d').Value := Trim(StringGrid1.Cells[6,Row]);
      Parameters.ParamByName('e').Value := StrToInt(StringGrid1.Cells[2,Row]);
      Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[3,Row]);
      Parameters.ParamByName('g').Value := StrToFloat(StringGrid1.Cells[4,Row]);
      Parameters.ParamByName('h').Value := Trim(StringGrid1.Cells[7,1]);
      ExecSQL;
    end;
    with Data.ADOQuery4 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Select * from 商品基础信息表 where 商品编号 = :a');
      Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,Row]);
      Open;
    end;
    Spjg := Data.ADOQuery4.FieldByName('进价').Value;
    with Data.ADOQuery1 do
    begin
      Close;
      SQL.Clear;//清空SQL语句
      //添加SQL语句
      SQL.Add('update 库存表 set 库存数量 = 库存数量 + :a ,库存金额 = 库存金额 + :b where
          商品编号 = :c');
      //为参数赋值
      Parameters.ParamByName('a').Value := StrToInt(StringGrid1.Cells[2,Row]);
      Parameters.ParamByName('b').Value := StrToInt(StringGrid1.Cells[2,Row])*Spjg;
      Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[0,Row]);
      ExecSQL;
    end;
  end;
```

```
    with Data.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(' update 客户结款表 set 欠款记账 = 欠款记账 - :b ,付款合计 = 付款合计 + :c ,
        结款时间 = Default where 客户编号 = :d');
      Parameters.ParamByName('b').Value :=StrToFloat(Label19.Caption)-
          StrToFloat(Edit2.Text);
      Parameters.ParamByName('c').Value :=  StrToFloat(Label19.Caption)-
          StrToFloat(Edit2.Text);
      Parameters.ParamByName('d').Value := Trim(StringGrid1.Cells[5,1]);
      ExecSQL;
    end;
    with Data.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('insert 客户结款记录表 values (:a ,:b ,:c ,Default,:d)');
      Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[5,1]);
      Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[6,1]);
      Parameters.ParamByName('c').Value := StrToFloat(Edit2.Text);
      Parameters.ParamByName('d').Value := Trim(Label21.Caption);
      ExecSQL;
    end;
    Data.ADOConnection1.CommitTrans;
    Application.MessageBox('保存成功。','提示',0+64);
    Clear;
    Except
      Data.ADOConnection1.RollbackTrans;
      Application.MessageBox('系统出错。','提示',64);
      Close;
    end;
  end;
```

18.8.3 客户订货管理设计

1. 概述

客户订货管理主要实现商品的预订功能。在进行订货处理时，程序需要记录客户编号、预订商品编号、商品数量、商品金额等信息。在程序运行中，当用户录入商品名称时，会以表格信息显示商品信息，以辅助用户录入。客户订货管理运行结果如图18.13所示。

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TStringGrid、TDataSource、TDBGrid、TListBox、TShape等组件。



图 18.13　客户订货运行结果

（2）设置各主要组件属性如表 18.10 所示。

表 18.10　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form27	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
ListBox1	TListBox	Visible	False
DBGrid1	TDBGrid	Visible	False
		DataSource	DataSource1
Edit8	TEdit	Text	1.0

3. 程序相关代码

单击“添加”按钮程序会自动生成订货票号。订货票号必须是惟一的，在生成订货票号时，是以固定字符串+系统日期字符串+序数编号组成的。首先指定一个固定字符串，例如“DP”，然后读取系统当前日期，将其转化为“yyyymmdd”格式的字符串，如果当前日期为 2005-11-25，则格式化后的字符串为“20051125”。最后是确定序数编号，首先查询最大的订单号，如果没有找到订单号，序数编号为“001”，否则提取订单号的序号。例如，订单号为“DP200506005”，提取的序号为“005”，然后将其转换为整数（例如 5），将该整数加 1，最后将其转换成新的序号（006）。在确定了固定字符串、系统日期字符串、序数编号之后，进行组合，生成新的订货票号（DP20051125006）。

```
procedure TForm22.BitBtn3Click(Sender: TObject);
var
   s,m: String;
   i: integer;
begin
   Label14.Caption := FormatDateTime('yyyy-mm-dd',Now());//取系统当前日期
   s:= 'DP'+FormatDateTime('yyyymmdd',Now());
```

```
    With Data.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select max(订单号) as ss From  订货票号表 ');
      Open;
    end;
    If Data.ADOQuery1.FieldByName('ss').Value = null then //是否有数据返回
      s := s + '001'
    else
    begin
      m:= Trim(Data.ADOQuery1.FieldByName('ss').Value) ;//提取序号
      i:= StrToInt(Trim(Copy(m,11,5))) ;//转换序号为整数
      if i<9 then
        s:= s + '00'+ InttoStr(i +1)
      else if i<99 then
        s:= s + '0'+ InttoStr(i +1)
      else
        s:= s +InttoStr(i+1);
    end;
    Label10.Caption := s;
    if Trim(Edit4.Text)='' then
      Edit4.SetFocus
    else
      Edit2.SetFocus;
  end;
```

下面是“保存”按钮的单击事件代码。客户订货主要记录订货的详细信息，涉及“订货票号表”和“客户订货表”两张数据表，直接向数据表中插入数据即可。为了减少操作过程中出现的错误，在保存订货信息时，首先检查数据是否有误。例如，订货信息是否有空项，在数据均符合要求后进行保存操作。

```
  procedure TForm22.BitBtn4Click(Sender: TObject);
  var
    i: Integer;
  begin   if Trim(Edit4.Text)<>'' then
    begin
      if JCxmlb = True then
      begin
        Data.ADOConnection1.BeginTrans;
        Try
        with Data.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('Insert  订货票号表 values (:a,:b,:c,:d,Default,:e,:f,Default)');
          Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[7,1]);
          Parameters.ParamByName('b').Value := StrToInt(Label15.Caption);
```

```
            Parameters.ParamByName('c').Value := StrToInt(Label16.Caption);
            Parameters.ParamByName('d').Value := StrToFloat(Label18.Caption);
            Parameters.ParamByName('e').Value := Trim(Edit4.Text);
            Parameters.ParamByName('f').Value := Trim(Label13.Caption);
            ExecSQL;
          end;
          For i := 1 to StringGrid1.RowCount-1 do
          begin
            with Data.ADOQuery1 do
            begin
              Close;
              SQL.Clear;
              SQL.Add('Insert 客户订货表 values (:a,:b,:c,:d,:e,:f,:g,:h)');
              Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[5,i]);
              Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[6,i]);
              Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[0,i]);
              Parameters.ParamByName('d').Value := Trim(StringGrid1.Cells[1,i]);
              Parameters.ParamByName('e').Value := StrToInt(StringGrid1.Cells[2,i]);
              Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[3,i]);
              Parameters.ParamByName('g').Value := StrToFloat(StringGrid1.Cells[4,i]);
              Parameters.ParamByName('h').Value := Trim(StringGrid1.Cells[7,1]);
              ExecSQL;
            end;
          end;
          Data.ADOConnection1.CommitTrans;
          Application.MessageBox('客户订货成功。','提示',64);
          Clear;
          BitBtn3.SetFocus;
          Except
          Data.ADOConnection1.RollbackTrans;
          Application.MessageBox('系统出错。','提示',64);
          Close;
          end;
        end
        else
          Application.MessageBox('项目列表有误！','提示',64);
      end
      Else
      begin
        Application.MessageBox('经手人不能为空。','提示',64);
        Edit4.SetFocus;
      end;
    end;
```

18.8.4 销售查询管理设计

1. 概述

销售查询主要实现对销售出去的商品按商品名称和日期进行查询。通过销售查询，管

理人员能够了解一段时间以来商品的销售情况，有利于进行决策分析。销售查询管理运行结果如图 18.14 所示。



图 18.14　销售查询运行结果

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TCheckBox、TDataTimePicker、TDataSource、TDBGrid 等组件。

（2）设置各主要组件属性如表 18.11 所示。

表 18.11　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form54	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
CheckBox1	TCheckBox	Checked	False
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

以下是“保存”按钮的单击事件代码。程序首先判断用户是否设置了查询条件，即“ComboBox1”或“ComboBox2”是否为空，如果为空，进行提示并退出查询，否则进一步确定是否进行时间段查询，即“CheckBox2”是否被选中，根据是否设置时间段查询执行不同的查询语句。

```
procedure TForm54.SpeedButton1Click(Sender: TObject);
begin
  if (Trim(ComboBox2.Text)<>'')and(Trim(ComboBox1.Text)<>'') then //是否设置查询条件
  begin
    CheckBox1.Checked := False;
    with Data.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      if CheckBox2.Checked = False then //不按时间段查询
      begin
        SQL.Add('select * from 销售表 where ');
```

```
            SQL.Add(Trim(ComboBox1.Text));
            SQL.Add('=');
            SQL.Add(':a');
          end
          else //按时间段查询
          begin
            SQL.Add('Select * from 销售表 where');
            SQL.Add(Trim(ComboBox1.Text));
            SQL.Add('=');
            SQL.Add(':a');
            SQL.Add('and 销售票号in (Select 销售票号from销售票号表where销售时间>=:b and销
            售时间<= :c)');
            Parameters.ParamByName('b').Value := Int(DateTimePicker1.Date)-2 ;
            Parameters.ParamByName('c').Value := Int(DateTimePicker2.Date)-2+ 0.9998;
          end;
          Parameters.ParamByName('a').value := Trim(ComboBox2.Text);
          Open;
        end;
        if Data.ADOQuery2.RecordCount >0 then
          DataSource1.DataSet := Data.ADOQuery2;
      end;
    end;
```

18.9 疑难问题分析解决

18.9.1 如何防止单击 TBitBtn 按钮时关闭模式窗体

在开发软件过程中，经常会遇到这种情况，如果窗体采用模式窗体，当用户单击 TBitBtn 按钮时，窗体可能会关闭。此时，需要修改 TBitBtn 的“Kind”属性为“bkCustom”，或者修改 TBitBtn 的“ModalResult”属性为“mrNone”，如图 18.15 所示。

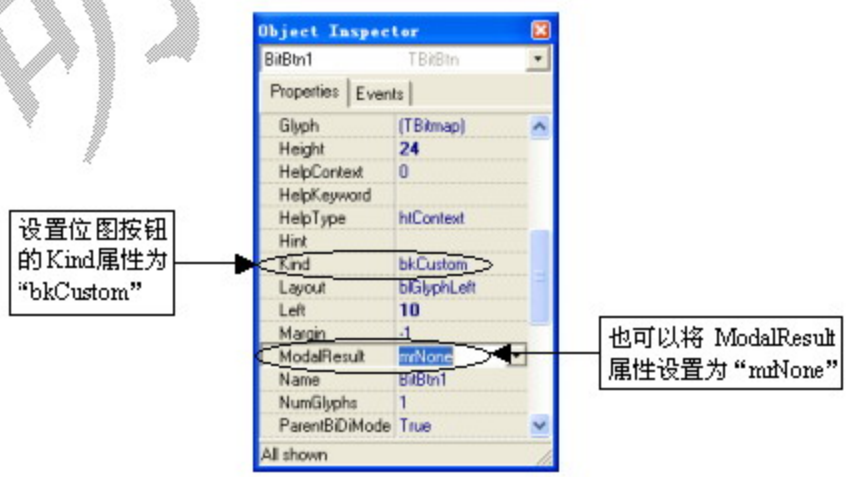


图 18.15 位图按钮设置

18.9.2 使用非数据控制组件显示和浏览数据

在开发数据库应用程序时，经常需要显示和浏览数据，使用数据控制组件显示、浏览数据虽然方便，但容易误删或修改数据，因此，在程序中经常使用 TEdit 等非数据控制组件显示、浏览数据。要实现非数据控制组件显示、浏览数据，需要自己编写相应的代码。首先需要从数据库中查询数据，将数据集当前记录数据赋给编辑框，在赋值过程中，需要注意类型转换。在浏览数据时，需要移动数据集当前指针，然后将数据集当前记录数据赋给编辑框。

18.9.3 在编辑框中只允许输入数字

用户在编辑框中录入数据时，有时只允许输入数字以防止用户输入非法数据。要实现此功能可以在编辑框的“OnKeyPress”事件中将一些按键屏蔽，代码如下：

```
procedure TForm7.Edit7KeyPress(Sender: TObject; var Key: Char);
var
  m: Boolean;
begin
  m := (key <#8)or(Key>#8)and(key <#46)or(key>#46)and(key <#48)or(key > #57);
  if m then
    key := #0;
end;
```

#8 为 BackSpace 键，#46 为小数点，#48~#57 为数字 0~9，#0 为空。

18.9.4 按 Enter 键控制焦点的移动

在开发程序时，为方便用户操作，需要控制控件焦点的移动。如果在每个控件的“OnKeyDown”事件中将控件的焦点移到另一个控件上，这样做可以实现焦点的转移，但并不是好的方法。因为在窗体中再放入一个控件时，可能需要修改许多控件的“OnKeyDown”事件代码。在 Delphi 中，提供了“FindNextControl”函数用来查找控件，该函数以 Tab 键顺序查找控件，其函数原型如下：

```
function FindNextControl(CurControl: TWinControl; GoForward, CheckTabStop, CheckParent:
    Boolean): TWinControl;
```

参数说明：

CurControl 代表当前的控件。

GoForward 表示查找方向，如果为“True”，向前查找，为“False”，向后查找。

CheckTabStop 与 CheckParent 确定是否对查找到的控件进行某种检查。如果 CheckTabStop 为“True”，则找到的控件的“TabStop”属性必须为“True”，否则函数不会将找到的控件作为返回值。如果 CheckParent 为“True”，则找到的控件的父类（Paraent 属性）必须是 Self，否则函数不会将找到的控件作为返回值。示例如下：

```
procedure TForm9.FindNext;
begin
  if FindNextControl(activeControl,True,False,False)is TEdit then
    TEdit(FindNextControl(activeControl,True,False,False)).SetFocus;
end;
procedure TForm9.Edit2KeyDown(Sender: TObject; var Key: Word;
```

```
    Shift: TShiftState); // 其他编辑框的OnKeyDown事件也调用了该方法
begin
  if Key = Vk_ReTurn then
    FindNext;
end;
```

18.9.5 在销售商品时，实现订单销售

在销售商品时，通常是临时销售商品，但有时也需要按订单销售。所以，在商品销售模块需要实现商品的临时销售和订单销售。商品的临时销售比较简单，用户输入客户名称，录入商品信息，单击“保存”按钮完成销售。商品的订单销售也比较简单，因为订单信息在商品订货时已经保存在数据库中，在按订单销售时，只需调出订单信息即可。但是，要实现商品临时销售与订单销售的整合相对比较困难。因为，在商品临时销售时，用户可以随意添加、删除商品，更改商品数量，但是，在按订单销售商品时，则不允许添加、删除商品，更不允许修改商品数量，而且在保存数据时，如果按订单销售，需要修改订单信息。在程序中，笔者使用了一个 Boolean 型的全局变量判断是否是按订单销售，如果为订单销售，不允许在表格中输入数据，在保存数据时，会进行额外的处理。

18.9.6 如何实现超期应收款查询

在本章的销售管理系统中，为了让用户能够了解一段时间以来没有结款的客户信息，设计了超期应收款查询模块。在程序中，允许用户输入超期天数，程序会以系统当前日期为基准减去超期天数，以该时间为条件从客户结款表中查询信息。代码如下：

```
with Data.ADOQuery1 do
begin
  Close;
  SQL.Clear;
  SQL.Add('select * from 客户结款表 where 结款时间 <= :a');
  Parameters.ParamByName('a').Value :=  Int(StrToDateTime(Edit1.Text))-
      StrToInt(Edit2.Text)+0.9998;
  Open;
end;
```

上面代码关键之处是查询条件“结款时间<=:a”，在客户的每一次结款时，都会在客户结款表中更改结款时间，因此，客户结款表中的结款时间是最近一次客户结款的时间。

18.10 程序设计清单

销售管理系统文件清单如表 18.12 所示。

表 18.12　　销售管理运行清单

文 件 名	文 件 类 型	说　明
Cqfkyj.dfm	窗体文件	超期应付款预警窗体
Datamodal.dfm	窗体文件	数据模块

续表

文 件 名	文 件 类 型	说 明
Dspxs.dfm	窗体文件	单商品销售报表窗体
Kcgl.dfm	窗体文件	库存管理窗体
Kexx1.dfm	窗体文件	显示客户信息窗体
Khdhxt.dfm	窗体文件	客户订货管理窗体
Khwldzd.dfm	窗体文件	客户往来对账单窗体
Khwljkgl.dfm	窗体文件	客户往来结款管理窗体
Khxx1.dfm	窗体文件	显示客户信息窗体
Khxx3.dfm	窗体文件	显示客户信息窗体
Khxxgl.dfm	窗体文件	客户基础信息管理窗体
Mp.dfm	窗体文件	模板窗体
Rxsbb.dfm	窗体文件	日销售报表窗体
Spjcxxgl.dfm	窗体文件	商品基础信息管理窗体
Spxscx.dfm	窗体文件	商品销售查询窗体
Spxsgl.dfm	窗体文件	商品销售管理窗体
Spxsthcx.dfm	窗体文件	商品销售退货查询窗体
Tjyusc.dfm	窗体文件	操作员管理窗体
Xsglxt.dfm	窗体文件	销售管理系统窗体
Xsthxt.dfm	窗体文件	销售退货管理窗体
Xsxx.dfm	窗体文件	显示客户信息窗体
Xtdl.dfm	窗体文件	系统登录窗体
Yxsbb.dfm	窗体文件	月销售报表窗体
Xs.dpr	工程文件	
Xs.exe	可执行文件	

18.11 技术、经验总结

18.11.1 技术总结

在 TStringGrid 中统计商品数量的实现

在商品销售管理模块中，用户输入商品数量后，程序会自动统计商品种类。请看下面的程序代码：

```
function TForm24.Kinds: Integer;
var
  i,m,mm: Integer;
  Iscf: Boolean;   //判断当前记录是否与当前行之前的记录重复
begin
  Result := StringGrid1.RowCount-1;//假设列表中商品没有重复，那么商品种类应为列表行数减1
  //双重循环的作用是列表中每一行数据与它后面的数据比较，如果数据重复，将Result减1
```

```
    For m := 1 to StringGrid1.RowCount-1 do   //第一层循环
      For i := m+1 to StringGrid1.RowCount-1 do //第二层循环
      begin
        Iscf := False;
        if m >1 then
          For mm := 1 to m-1 do   //将当前行数据与当前行之前的记录比较，如果重复，继续第二层
          //循环，Result值不变
            if Trim(StringGrid1.Cells[0,m])= Trim(StringGrid1.Cells[0,mm])then
            begin
              Iscf := True;
              Break;
            end;
        if Iscf= True then
          Continue;
        if Trim(StringGrid1.Cells[0,m])=Trim(StringGrid1.Cells[0,i])then
          ReSult := Result -1;
      end;
  end;
```

18.11.2 经验总结

在开发数据库应用程序时，经常需要通过程序向数据库中录入数据。例如，销售商品时，需要将销售信息保存到数据表中。当保存数据后，要注意清空界面上用户录入的数据，这样做有利于进行下一次操作，而且会防止用户不小心向数据表中录入重复数据。笔者建议写一个单独的过程或函数用于清空用户录入的数据。在窗体显示时，可以调用该过程或函数进行数据初始化，当用户单击“取消”按钮时，也将调用该过程或函数进行取消操作。

18.12 系统的编译与发行

本系统编译生成的可执行文件为1.81MB，如果要减小可执行文件的大小，可以进行压缩。要在其他机器上运行本系统，需要具备以下条件。

- 具有本系统的可执行文件。
- 将数据库“销售管理系统”附加到SQL Server 2000中。

由于本系统使用的是ADO控件连接数据库，所以不需要配置BDE或ODBC。要把本系统做成安装文件，可以参考第16章16.13小节“程序的打包与发行”。

第19章 Delphi在工资管理系统中的应用

19.1 工资管理系统概述

19.1.1 系统任务

工资管理系统是现代企事业、行政、科研单位必不可少的管理软件。它不仅能进行准确的工资核算，而且能够方便地进行工资发放，因此，工资管理系统的任务是使用计算机对员工工资信息进行日常管理。

19.1.2 设计目标

- 实现员工信息的添加、修改、删除
- 实现员工工资信息的添加、修改、删除
- 实现工资发放
- 实现工资调整管理
- 实现工作条的打印

19.2 需求分析

19.2.1 业务调查

松原师范学院是一所培养小学教师的专业学校。该院共有教职工300多人，学生7000多人。受社会信息化发展和教育体制改革的影响，该院原始的管理方式暴露出种种弊端，尤其在教师工资管理中，由于新的教师工资制度的实施，工资核算经常出错。经学校领导研究决定，使用计算机进行教师工资核算、发放。

19.2.2 系统规划

针对上面工资管理的具体情况，定义如下需求分析。

1. 系统目标

利用计算机系统资源帮助财会人员完成工资管理。

2. 系统处理范围

- 员工信息维护
- 工资管理
- 报表打印

3. 系统功能

- 员工信息添加、删除
- 操作员管理
- 员工基本工资信息管理

在员工基本工资管理中主要保存员工固定工资，例如，基本工资、养老保险金等。

- 员工浮动工资信息管理

在员工浮动工资管理中主要保存员工经常变化的工资，例如，加班工资、事假扣款等。

- 员工工资发放
- 打印工资条

4. 系统结构图

工资管理系统结构图如图 19.1 所示。

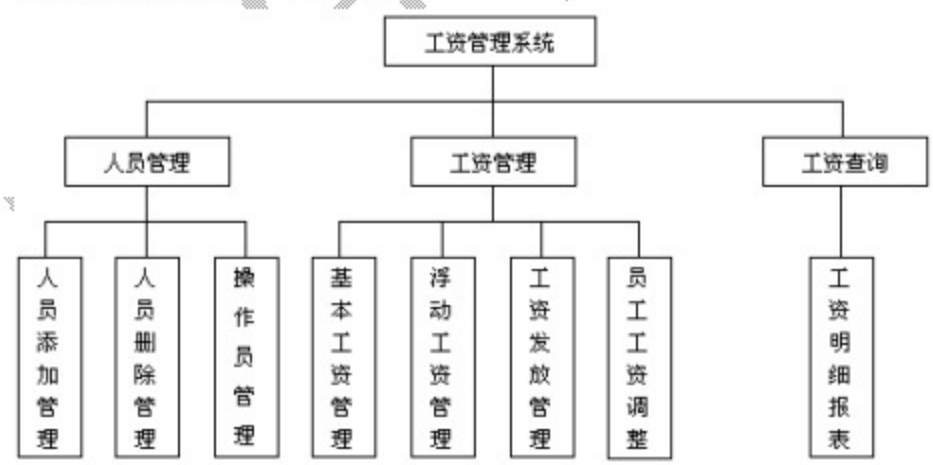


图 19.1 工资管理系统结构图

19.3 数据库设计

19.3.1 数据库概念设计

工资管理系统的 E-R 图如图 19.2 所示。



图 19.2　工资管理系统 E-R 图

19.3.2　数据库逻辑结构设计

配送管理系统共使用了 6 张数据表，分别为：操作员信息表、员工基础信息表、工资发放明细表、工资发放（视图）、员工基本工资表、员工浮动工资项目表。由于篇幅关系，在此只列出使用较多的数据表结构（表 19.1～表 19.4），其他表结构请参考光盘。

表 19.1　　员工基本工资表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
员工编号	Varchar(30)			否	
员工姓名	Varchar(30)				
基本工资	Money				
职务工资	Money				
工龄工资	Money				
养老保险金	Money				

表 19.2　　员工浮动工资项目表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
员工编号	Varchar(30)				
员工姓名	Varchar(30)				
个人所得税	Money				
住房补助	Money				
保险费	Money				
生活补助	Money				

表 19.3　　工资发放

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
员工编号	Varchar(30)				
员工姓名	Varchar(30)				
基本工资	Money				
职务工资	Money				
工龄工资	Money				
养老保险金	Money				
个人所得税	Money				
住房补助	Money				
保险费	Money				
生活补助	Money				

表 19.4　　工资发放明细表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
员工编号	Varchar(30)				
员工姓名	Varchar(30)				
基本工资	Money				
职务工资	Money				
工龄工资	Money				
养老保险金	Money				
加班天数	Int				
加班工资	Money				
事假天数	Int				
事假工资	Money				
个人所得税	Money				
住房补助	Money				
保险费	Money				
生活补助	Money				
应发金额	Money				
实发金额	Money				
余额	Money				
时间	DateTime				GetDate()

19.3.3　数据字典

如表 19.5 和表 19.6 所示。

表 19.5　　员工基本工资表

字 段 名 称	描　　述	说　　明
员工编号	标识一个员工	用来区分同名的员工
员工姓名	标记员工名称	

表 19.6　　工资发放明细表

字 段 名 称	描　　述	说　　明
加班天数	记录本月员工加班天数	
加班工资	记录本月加班工资	
事假天数	记录本月员工请假天数	
事假工资	记录本月员工事假扣款	
应发金额	记录本月应发工资	
实发金额	记录本月实发工资	
余额	记录员工本月余额	如果为正数表示欠员工钱，为负数表示员工欠单位钱

19.4 处理过程分析

19.4.1 工资发放处理过程分析

财会人员在进行工资发放时，需要显示职工工资信息，以便于核算工资。所以，首先应输入职工名称，由程序调出职工工资信息，然后输入加班天数、事假天数，程序会核算加班工资、事假扣款、应发工资，最后得出实发工资，将职工工资信息保存到数据库中。工资发放管理业务流程如图 19.3 所示。

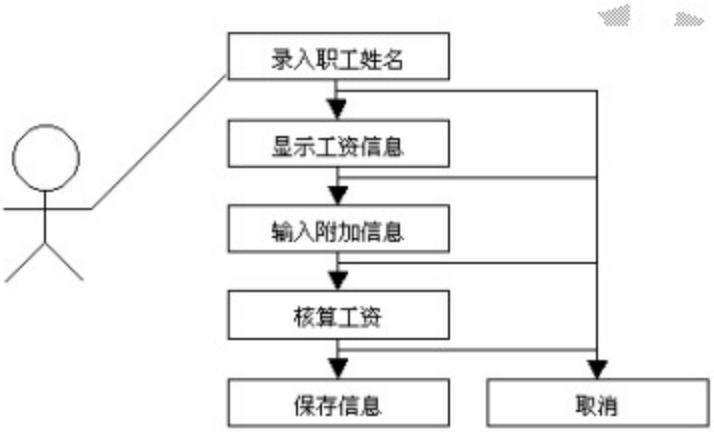


图 19.3 工资发放业务流程图

19.4.2 工资调整处理过程分析

调整职工工资，首先需要将职工工资信息查询出来，然后由操作员输入新的职工工资信息，最后保存信息。需要注意的是，有时可能只需修改职工部分工资信息，例如，只调整基本工资，此时，只需在基本工资编辑框中输入数据即可。

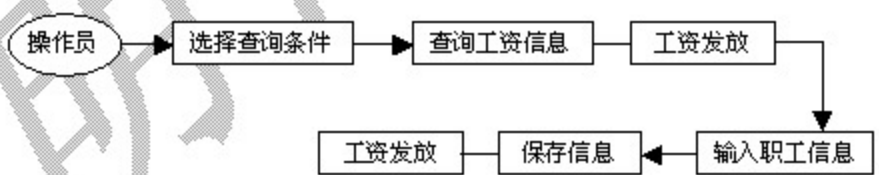


图 19.4 工资调整开发流程

19.5 主窗体设计

主窗体设计步骤如下。

（1）创建一个窗体，在窗体上放置 TMainMenu、TImageList、TStatusBar、TImage 等组件，如图 19.5 所示。

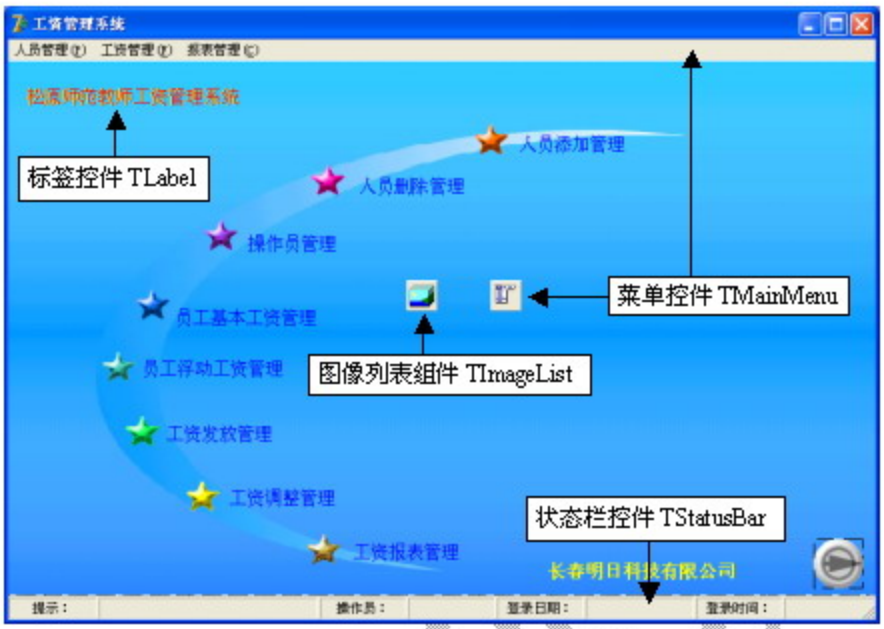


图 19.5　主窗体界面

（2）设置各主要组件属性如表 19.7 所示。

表 19.7　　组件属性设置

对　象　名	对 象 类 型	属　　性	值
Form1	Tform	Font.Color	clBlue
		Font.Size	8
MainMenu1	TMainMenu	Images	ImageList1

19.6　菜 单 设 计

菜单设计步骤如下。

（1）在窗体中选中 MainMenu1 组件，在对象观察器中设置“Images”属性为“ImageList1”。

（2）双击 MainMenu1 组件，打开菜单设计器，效果如图 19.6 所示。

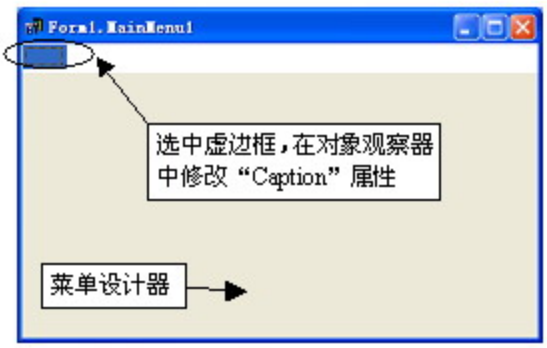


图 19.6　菜单设计器

（3）在菜单设计器中有一个虚边框，选中该虚边框，在对象观察器中设置“Caption”属性，例如，“人员管理(&P)”，效果如图 19.7 所示。

（4）如果实现菜单的分组，可以在对象观察器中设置“Caption”属性为“-”。按照上面的方法添加其他菜单项，效果如图 19.8 所示。

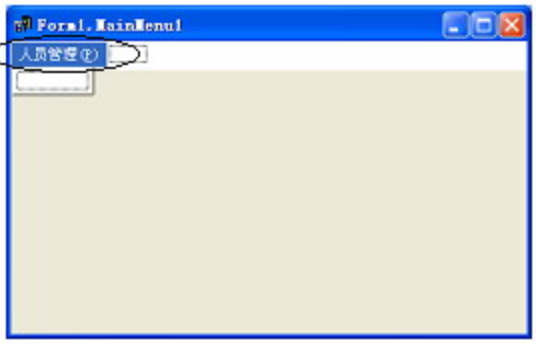


图 19.7　菜单效果图



图 19.8　系统菜单

19.7　数据模块设计

数据模块设计步骤如下。

（1）启动 Dephi 7，打开工程“gzgl”，单击“File”/“New”/“Data Module”菜单项，创建数据模块。

（2）选中数据模块，在对象观察器中修改“Name”属性为“Data3”。

（3）在数据模块上放置 TADOConnection 组件、TADOQuery 组件，如图 19.9 所示。

（4）设置各主要组件属性如表 19.8 所示。

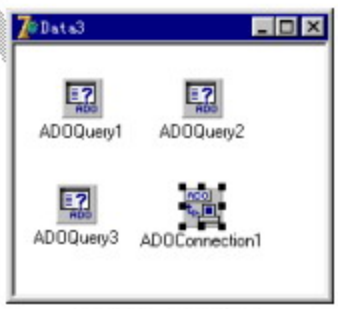


图 19.9　数据模块

表 19.8　　组件属性设置

对 象 名	对 象 类 型	属　性	值
ADOConnection1	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=工资管理系统
ADOQuery1	TADOQuery	Connection	ADOConnection1
ADOQuery2	TADOQuery	Connection	ADOConnection1
ADOQuery3	TADOQuery	Connection	ADOConnection1

（5）单击 Save All 按钮，将单元保存为“Data”。

19.8　窗体模板设计

1. 概述

在开发 Windows 应用程序时，经常需要创建多个窗体，为了美化界面，一般会在窗体

上放置面板等组件，程序员会为此做许多重复工作。如果程序中的一些窗体有共性，例如，窗体上都包含面板组件，可以将这些窗体的共性抽取出来，创建一个父窗体。例如，在父窗体上放置一个面板组件，将其他窗体派生于父窗体，这样，在子窗体中便会有一个面板组件。

2. 设计步骤

（1）新建一个窗体，在窗体上放置一个 TPanel 组件。

（2）设置各主要组件属性如表 19.9 所示。

表 19.9　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form41	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
asa	TPanel	Align	alClient

19.9　程序设计与编码

19.9.1　员工基本工资管理设计

1. 概述

员工基本工资管理主要是保存员工固定工资信息，例如，员工基本工资、职务工资、养老保险金等。操作员在列表中输入员工名称，按 Enter 键会将员工编号显示出来。如果想删除当前行可以按 Delete 键，添加新行可以按“↓”方向键，前提是当前行数据不能为空。员工基本工资管理运行结果如图 19.10 所示。



图 19.10　基本工资管理运行界面

2. 设计步骤

（1）打开工程“gzgl”，单击“File”/“New”/“Others”菜单项，打开一个对话框，打开“gzgl”选项卡，如图 19.11 所示。

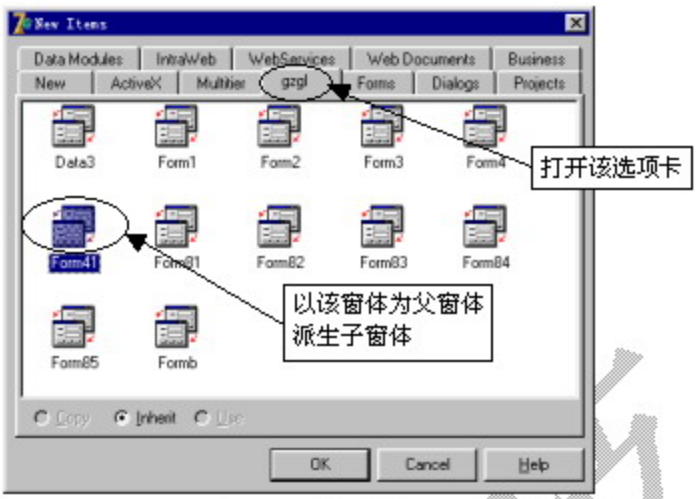


图 19.11　新建项目

（2）选中“Form41”选项，单击 OK 按钮，创建窗体。

（3）在窗体上放置 TBitBtn、TLabel、TStringGrid、TListBox 等组件。

（4）设置各主要组件属性，如表 19.10 所示。

表 19.10　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form81	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
BitBtn3	TBitBtn	Kind	bkClose

3. 程序相关代码

```
TForm81 = class(TForm41) //TForm81是从TForm41继承而来的
var
  Form81: TForm81;
  x,y: Integer;
implementation
  uses Dataa;
{$R *.dfm}
```

处理 StringGrid1 单元格被选中时的事件，记录当前单元格坐标，判断 StringGrid1 是否可以编辑。当单元格处于“员工姓名”列，并且“员工编号”为空时可以编辑数据，或者当前单元格处于“基本工资”、“职务工资”、“工龄工资”、“养老保险金”时，“员工编号”不为空，可以编辑数据，其他情况不可以编辑数据。

```
procedure TForm81.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  inherited;
  y = ACol;
```

```
    x := ARow;
    //确定单元格可以编辑的条件
    if (ACol = 1)and( Trim(StringGrid1.Cells[0,x])='')or(((ACol = 2)or(Acol = 3)or(Acol =
    4)or(Acol = 5))and(Trim(StringGrid1.Cells[0,x])<>'')) then
      StringGrid1.Options := StringGrid1.Options +[goEditing]
    else
      StringGrid1.Options := StringGrid1.Options -[goEditing];
  end;
```

自定义函数 IsNull 判断列表中当前行数据是否为空，如果为空，函数返回值为 True，否则为 False。

```
function TForm81.IsNull: Boolean;
var
  a: Integer;
begin
  IsNull := False;
  For a:= 0 to StringGrid1.ColCount -1 do
  begin
    if Trim(Stringgrid1.Cells[a,x])='' then
    begin
      IsNull := True;
      Break;
    end;
  end;
end;
```

处理 StringGrid1 的 OnKeyDown 事件，为了增加数据录入的灵活性，操作员在录入数据过程中可以删除整行数据（按 Delete 键），也可以添加新数据（按下“↓”方向键）。

在删除数据时，首先判断表格中的数据是否超过两行（首行为标题），如果超过两行，提示是否删除数据，如果用户选择“是”，程序将当前行之后的数据赋给其上一行数据，然后清空并删除最后一行数据。如果表格中数据等于两行，则清空当前行数据（第 2 行数据，首行数据为标题，始终不会被选中）。

在添加数据时，如果当前表格中没有为空的数据，并且当前行处于最后一行，按下“↓”方向键将添加新行。

```
procedure TForm81.StringGrid1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
  R,L: Integer;
begin
  if Key = Vk_Delete then
  begin
    if StringGrid1.RowCount>2 then
    begin
      If Application.MessageBox('确实要删除该条记录吗？','提示',MB_YESNO )= ID_Yes then
      begin
        ClearString;//清空当前行数据
```

```
        if x <> StringGrid1.RowCount-1 then
        begin
          For r := x+1 to StringGrid1.RowCount-1 do //利用双重循环将当前行之后的数据赋给
          //上一行
            For l := 0 to StringGrid1.ColCount-1 do
              StringGrid1.Cells[l,r-1]:= StringGrid1.Cells[l,r];
        end;
        ClearStringEnd;//清空最后一行数据
        StringGrid1.RowCount := StringGrid1.RowCount-1;
        StringGrid1.SetFocus;
        StringGrid1.Col := 2;
        StringGrid1.Col := 1;
        Exit;
      end;
    end
    else if StringGrid1.RowCount = 2 then
      If Application.MessageBox('确实要删除该条记录吗？','提示',MB_YESNO )= ID_Yes then
      begin
        ClearString;
        StringGrid1.SetFocus;
        StringGrid1.Col := 2;
        StringGrid1.Col := 1;
      end;
  end;
  if (Key = Vk_Down)and(IsNull = False)and(x = StringGrid1.RowCount-1) then
  begin
    StringGrid1.RowCount := StringGrid1.RowCount +1;
    StringGrid1.Col := 1;
  end
  else if (Key = Vk_ReTurn)and(Trim(StringGrid1.Cells[0,x])='')and
    (Trim(StringGrid1.Cells[1,x])<>'')then
  begin
    with Data3.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Select * from 员工基础信息表 where 员工姓名 = :a');
      Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[1,x]);
      Open;
    end;
    if Data3.ADOQuery1.RecordCount >0 then
      if Data3.ADOQuery1.RecordCount>1 then
      begin
        ListBox1.Clear;
        while Not Data3.ADOQuery1.Eof  do
        begin
          ListBox1.Items.Add(Data3.ADOQuery1.FieldByName('员工编号').Value);
```

```
            Data3.ADOQuery1.Next ;
          end;
          ListBox1.Visible := True;
          ListBox1.SetFocus;
        end
        else
        begin
          StringGrid1.Cells[0,x]:= Data3.ADOQuery1.FieldByName('员工编号').Value;
          StringGrid1.Col := 2;
        end
      else
      begin
        StringGrid1.Cells[1,x]:= '';
        Application.MessageBox('该员工不存在，请重新输入。','提示',64);
      end;
    end
    else if (Key = VK_ReTurn)and(Trim(StringGrid1.Cells[0,x])<>'')then
    begin
      if StringGrid1.Col < StringGrid1.ColCount-1 then
        StringGrid1.Col := StringGrid1.Col+1;
    end;
    if (Key = VK_Shift)and(JCxmlb = True) then
      BitBtn1.SetFocus;
  end;
```

以下是“保存”按钮的单击事件代码，单击“保存”按钮，会将列表中的数据保存到数据库中。在保存时，首先检查数据是否有错（数据是否完整），如果有误，进行提示，并退出操作；否则检查员工信息是否重复，如果表格中的员工信息重复，进行提示，并退出操作。在表格中数据均正确的情况下，利用循环保存数据，将数据插入到“员工基本工资表”中。

```
procedure TForm81.BitBtn1Click(Sender: TObject);
var
  i: integer;
begin
  inherited;
  if Jcxmlb = True then
  begin
    if iscf = True then
    begin
      Application.MessageBox('员工信息不能重复。','提示',64);
      Exit;
    end;
    if Biscf = False then
    begin
      Try
        For i := 1 to StringGrid1.RowCount-1 do
        begin
          with Data3.ADOQuery2 do
```

```
                begin
                  Close;
                  SQL.Clear;
                  SQL.Add('insert 员工基本工资表 values(:a,:b,:c,:d,:e,:f)');
                  Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,i]);
                  Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[1,i]);
                  Parameters.ParamByName('c').Value := StrToFloat(StringGrid1.Cells[2,i]);
                  Parameters.ParamByName('d').Value := StrToFloat(StringGrid1.Cells[3,i]);
                  Parameters.ParamByName('e').Value := StrToFloat(StringGrid1.Cells[4,i]);
                  Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[5,i]);
                  ExecSQL;
                end;
              end;
              Application.MessageBox('操作成功。','提示',64);
              Clear;
            Except
              Application.MessageBox('系统出错。','提示',64);
              Close;
            end;
          end;
        end
        else
          Application.MessageBox('项目列表有误。','提示',64);
    end;
```

19.9.2 员工浮动工资管理设计

1. 概述

浮动工资是指员工经常变动的工资，例如，个人所得税、生活补助、保险费等。员工浮动工资管理主要是保存员工浮动工资信息。为了帮助操作员实现员工信息快速录入，在窗体显示时，会将所有需要添加浮动工资的员工信息显示出来。员工浮动工资管理运行结果如图 19.12 所示。



图 19.12 浮动工资管理运行界面

2. 设计步骤

（1）新建一个继承于 TForm41 父窗体的子窗体，在窗体上放置 TBitBtn、TLabel、

TStringGrid、TPaenl等组件。

（2）设置各主要组件属性，如表19.11所示。

表19.11　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form82	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
BitBtn3	TBitBtn	Kind	bkClose

3. 程序相关代码

处理StringGrid1的OnKeyPress事件，防止输入非法字符。在当前单元格处于“生活补助”、“住房补助”、“保险费”、“个人所得税”列时，将限制数据的录入，只允许输入数字。

```
procedure TForm82.StringGrid1KeyPress(Sender: TObject; var Key: Char);
var
  mm: Boolean;
begin
  //防止在StringGrid1中输入非法字符
  if(y = 2)or(y =3)or(Y = 4)or(y = 5)then
  begin
    mm := (Key <#8)or(Key >#8)and(Key <#46)or(Key>#46)and(Key<#48)or(Key>#57);
    if mm then
      Key := #0;
  end;
end;
```

以下是“保存”按钮的事件代码，程序首先检查列表中的数据是否有误（数据是否为空、是否有重复数据），如果没有错误，将利用循环将数据保存到“员工浮动工资项目表”中。

```
procedure TForm82.BitBtn1Click(Sender: TObject);
var
  i: integer;
begin
  inherited;
  if Jcxmlb = True then
  begin
    if iscf = True then
    begin
      Application.MessageBox('员工信息不能重复。','提示',64);
      Exit;
    end;
    if Biscf = False then
    begin
      Try
        For i := 1 to StringGrid1.RowCount-1 do
        begin
```

```
                with Data3.ADOQuery2 do
                begin
                  Close;
                  SQL.Clear;
                  SQL.Add('insert  员工浮动工资项目表  values(:a ,:b,:c,:d,:e,:f)');
                  Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,i]);
                  Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[1,i]);
                  Parameters.ParamByName('c').Value := StrToFloat(StringGrid1.Cells[5,i]);
                  Parameters.ParamByName('d').Value := StrToFloat(StringGrid1.Cells[2,i]);
                  Parameters.ParamByName('e').Value := StrToFloat(StringGrid1.Cells[4,i]);
                  Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[3,i]);
                  ExecSQL;
                end;
              end;
              Application.MessageBox('操作成功。','提示',64);
              Clear;
            Except
              Application.MessageBox('系统出错。','提示',64);
              Close;
            end;
          end;
        end
        else
          Application.MessageBox('项目列表有误。','提示',64);
    end;
```

19.9.3 工资发放管理设计

1. 概述

工资发放管理主要是帮助财会人员完成员工的工资发放。财会人员录入员工姓名后，程序会调出该员工的基本工资、浮动工资信息。为了实现员工工资的快速发放，在窗体显示时，会将所有需要发放工资的员工工资信息显示出来，如果操作员只需对部分员工进行工资发放，可以按 Delete 键将其他员工信息从表格中删除。工资发放管理运行结果如图 19.13 所示。

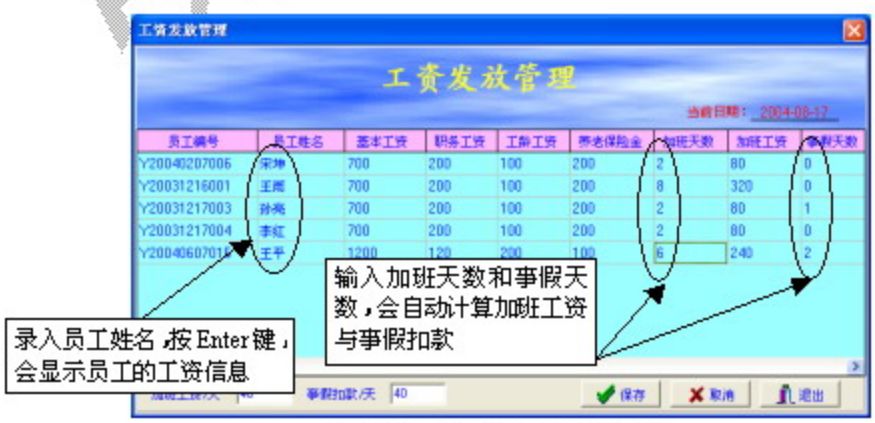


图 19.13 工资发放管理运行界面

2. 设计步骤

（1）新建一个继承于 TForm41 父窗体的子窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TStringGrid、TShape 等组件。

（2）设置各主要组件属性如表 19.12 所示。

表 19.12　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form83	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
BitBtn3	TBitBtn	Kind	bkClose

3. 程序相关代码

处理用户编辑 StringGrid1 中的文本时触发的事件，当用户输入加班天数和事假天数时，程序会自动计算加班工资和事假扣款。

```
procedure TForm83.StringGrid1SetEditText(Sender: TObject; ACol,
  ARow: Integer; const Value: String);
begin
  inherited;
  if (y = 6) or (y =8) then
  begin
    if Trim(StringGrid1.Cells[6,x])<>'' then
    begin
      StringGrid1.Cells[7,x]:= FloatToStr(StrToInt(StringGrid1.Cells[6,x])
      *StrToFloat(Edit1.Text)) ;
       if (Trim(StringGrid1.Cells[0,x])<>'')and(Trim(StringGrid1.Cells[9,x])<>'')then
         StringGrid1.Cells[15,x]:= FloatToStr(StrToFloat(StringGrid1.Cells[2,x])
         +StrToFloat(StringGrid1.Cells[3,x])+StrToFloat(StringGrid1.Cells[4,x])
         -StrToFloat(StringGrid1.Cells[5,x])+StrToFloat(StringGrid1.Cells[7,x])
         -StrToFloat(StringGrid1.Cells[9,x])-StrToFloat(StringGrid1.Cells[10,x])
         +StrToFloat(StringGrid1.Cells[11,x])+StrToFloat(StringGrid1.Cells[12,x])
         +StrToFloat(StringGrid1.Cells[13,x])+StrToFloat(StringGrid1.Cells[14,x]));
    end
    else
    begin
      StringGrid1.Cells[7,x]:= '';
      StringGrid1.Cells[15,x]:='';
    end;
    if Trim(StringGrid1.Cells[8,x])<>'' then
    begin
      StringGrid1.Cells[9,x]:= FloatToStr(StrToInt(StringGrid1.Cells[8,x])
        *StrToFloat(Edit2.Text));
      if (Trim(StringGrid1.Cells[0,x])<>'')and(Trim(StringGrid1.Cells[7,x])<>'')then
        StringGrid1.Cells[15,x]:= FloatToStr(StrToFloat(StringGrid1.Cells[2,x])
```

```
            +StrToFloat(StringGrid1.Cells[3,x])+StrToFloat(StringGrid1.Cells[4,x])
            -StrToFloat(StringGrid1.Cells[5,x])+StrToFloat(StringGrid1.Cells[7,x])
            -StrToFloat(StringGrid1.Cells[9,x])-StrToFloat(StringGrid1.Cells[10,x])
            +StrToFloat(StringGrid1.Cells[11,x])+StrToFloat(StringGrid1.Cells[12,x])
            +StrToFloat(StringGrid1.Cells[13,x])+StrToFloat(StringGrid1.Cells[14,x]));
      end
      else
      Begin
        StringGrid1.Cells[9,x]:='';
        StringGrid1.Cells[15,x]:='';
      end ;
    end;
end;
```

以下是“保存”按钮的事件代码，程序首先检查列表中是否有误，即是否有空项，职工信息是否有重复，如果职工工资信息无误，利用循环将这些信息保存到数据表“工资发放明细表”中。

```
procedure TForm83.BitBtn1Click(Sender: TObject);
var
  i: integer;
begin
  inherited;
  if Jcxmlb = True then //检查数据是否为空
  begin
    if iscf = True then //检查数据是否重复
    begin
      Application.MessageBox('员工信息不能重复。','提示',64);
      Exit;
    end;
    if Biscf = False then
    begin
      Try
        For i := 1 to StringGrid1.RowCount-1 do
        begin
          with Data3.ADOQuery2 do
          begin
            Close;
            SQL.Clear;
            SQL.Add('insert 工资发放明细表 values(:a ,:b,:c,:d,:e,:f,:g,:h, :i,:j,:k, :l,
              :m,:n,:o,:p,Default)');
            Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[0,i]);
            Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[1,i]);
            Parameters.ParamByName('c').Value := StrToFloat(StringGrid1.Cells[2,i]);
            Parameters.ParamByName('d').Value := StrToFloat(StringGrid1.Cells[3,i]);
            Parameters.ParamByName('e').Value := StrToFloat(StringGrid1.Cells[4,i]);
            Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[5,i]);
            Parameters.ParamByName('g').Value := StrToInt(StringGrid1.Cells[6,i]);
```

```
                Parameters.ParamByName('h').Value := StrToFloat(StringGrid1.Cells[7,i]);
                Parameters.ParamByName('i').Value := StrToInt(StringGrid1.Cells[8,i]);
                Parameters.ParamByName('j').Value := StrToFloat(StringGrid1.Cells[9,i]);
                Parameters.ParamByName('k').Value := StrToFloat(StringGrid1.Cells[10,i]);
                Parameters.ParamByName('l').Value := StrToFloat(StringGrid1.Cells[11,i]);
                Parameters.ParamByName('m').Value := StrToFloat(StringGrid1.Cells[13,i]);
                Parameters.ParamByName('n').Value := StrToFloat(StringGrid1.Cells[14,i]);
                Parameters.ParamByName('o').Value := StrToFloat(StringGrid1.Cells[15,i]);
                Parameters.ParamByName('p').Value := StrToFloat(StringGrid1.Cells[16,i]);
                ExecSQL;
              end;
            end;
            Application.MessageBox('操作成功。','提示',64);
            Clear;
          Except
            Application.MessageBox('系统出错。','提示',64);
            Close;
          end;
        end;
      end
      else
        Application.MessageBox('项目列表有误。','提示',64);
end;
```

19.9.4 员工工资调整管理设计

1. 概述

工资调整是工资管理系统必备的功能，由于国家政策、企业效益、个人工作情况等几方面因素的影响，企业员工工资不是一成不变的。为了适应企事业需求，在工资管理系统中都设有工资调整。员工工资调整运行结果如图 19.14 所示。



图 19.14 员工工资调整

2. 设计步骤

（1）新建一个继承于 TForm41 父窗体的子窗体，在窗体上放置 TPanel、TBitBtn、

TLabel、TComboBox、TEdit、TDataSource、TDBGrid等组件。

（2）设置各主要组件属性，如表19.13所示。

表19.13　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form84	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
BitBtn3	TBitBtn	Kind	bkClose
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

Clear过程主要用于进行初始化工作，清空控件中的文本。

```
procedure TForm84.Clear;
begin
  ComboBox1.Text := '';
  ComboBox2.Clear;
  DataSource1.DataSet := Nil;
  Edit1.Clear;
  Edit2.Clear;
  Edit3.Clear;
  Edit4.Clear;
  Edit5.Clear;
  Edit6.Clear;
  Edit7.Clear;
  Edit8.Clear;
  BitBtn1.Enabled := False;
end;
```

处理ComboBox1的OnChange事件，当ComboBox1当前内容改变时，ComboBox2中的数据会跟着相应地改变，方便用户设置查询条件。

```
procedure TForm84.ComboBox1Change(Sender: TObject);
begin
  if Trim(ComboBox1.Text)<>'' then
  begin
    with Data3.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Select Distinct ');
      SQL.Add(Trim(ComboBox1.Text));
      SQL.Add('From 工资发放');
      Open;
    end;
    if Data3.ADOQuery1.RecordCount>0 then
```

```
      begin
        ComboBox2.Clear;
        While Not Data3.ADOQuery1.Eof do
        begin
          ComboBox2.Items.Add(Data3.ADOQuery1.Fields[0].Value);
          Data3.ADOQuery1.Next;
        end;
      end;
    end;
end;
```

下面是“查询”按钮的单击事件代码。程序首先检查用户是否设置了查询条件，即“ComboBox1”或“ComboBox2”是否为空，如果为空，退出查询操作；如果不为空，根据查询条件从“工资发放”数据表中查询数据。

```
procedure TForm84.SpeedButton1Click(Sender: TObject);
begin
  if (Trim(ComboBox1.Text)<>'')and(Trim(ComboBox2.Text)<>'')then //是否设置了查询条件
  begin
    with Data3.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 工资发放 where');
      SQL.Add(Trim(ComboBox1.Text)) ;
      SQL.Add('= :a');
      Parameters.ParamByName('a').Value := Trim(ComboBox2.Text);
      Open;
    end;
    if Data3.ADOQuery2.RecordCount >0 then
    begin
      DataSource1.DataSet := Data3.ADOQuery2;
      EditValue;
      BitBtn1.Enabled := True;
    end
    else
    begin
      DataSource1.DataSet := Nil;
      BitBtn1.Enabled := False;
      Application.MessageBox('该员工信息不全或不存在。','提示',64);
    end;
  end;
end;
```

以下是“保存”按钮的单击事件代码，在保存修改的员工工资信息时，程序会提示是否进行保存，防止操作员意外地修改数据，如果用户选择“是”，程序会检查当前员工信息是否完整，如果不完整，退出保存操作；如果信息完整，则修改“工资发放”视图完成工资调整。

```
procedure TForm84.BitBtn1Click(Sender: TObject);
begin
  if Application.MessageBox(Pchar('确实要修改员工编号为'+Trim(Data3.ADOQuery2.FieldByName
('员工编号').Value)+'的员工数据吗？'),'提示',MB_YesNo)= ID_Yes then
  begin
    if EditIsNull = False then
    begin
      Try
        Data3.ADOConnection1.BeginTrans;//开始一个事务
        with Data3.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('update 工资发放 set 基本工资 = :a,职务工资 = :b,工龄工资 = :c,养老保险
          金 = :d where 员工编号 = :e');
          Parameters.ParamByName('a').Value := StrToFloat(Edit1.Text);
          parameters.ParamByName('b').Value := StrToFloat(Edit2.Text);
          parameters.ParamByName('c').Value := StrToFloat(Edit3.Text);
          parameters.ParamByName('d').Value := StrToFloat(Edit4.Text);
          Parameters.ParamByName('e').Value := Trim(Data3.ADOQuery2.FieldByName('员工编号
          ').Value);
          ExecSQL;
        end;
        with Data3.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('update 工资发放 set 住房补助 = :a,生活补助 = :b,保险费 = :c,个人所得税
          = :d where 员工编号 = :e');
          Parameters.ParamByName('a').Value := StrToFloat(Edit6.Text);
          parameters.ParamByName('b').Value := StrToFloat(Edit7.Text);
          parameters.ParamByName('c').Value := StrToFloat(Edit8.Text);
          parameters.ParamByName('d').Value := StrToFloat(Edit5.Text);
          Parameters.ParamByName('e').Value := Trim(Data3.ADOQuery2.FieldByName('员工编号
          ').Value);
          ExecSQL;
        end;
        Data3.ADOConnection1.CommitTrans;//提交事务
        Application.MessageBox('修改成功.','提示',64);
        Clear;
      Except
        Data3.ADOConnection1.RollbackTrans;//回滚事务
        Application.MessageBox('系统出错.','提示',64);
      End;
    end;
  end;
end;
```

19.10 疑难问题分析解决

19.10.1 使用 TRVSystem 组件编写报表

TRVSystem 是 Delphi 7 新添加的用来编写报表的组件。通过该组件，程序开发人员可以编写各种复杂的报表。TRVSystem 组件的常用属性、事件如下。

- GridHoriz 属性

定义预览屏幕水平方向的网格间距，默认值为 0.0，表示无网格。

- OutputFileName 属性

指定保存输出报表的文件名。

- TitlePreview 属性

定义报表预览窗体的标题。

- TitleSetup 属性

定义报表 Setup 窗体的标题。

- OnPreviewShow 事件

在预览窗体的 OnShow 事件期间触发该事件。

- OnPrint 事件

打印报表主体时触发该事件，完成报表打印后退出该事件。

在使用 TRVSystem 组件编写报表时，通常是在其 OnPrint 事件中编写代码，示例如下：

```
procedure TForm85.aaPrint(Sender: TObject);
begin
  with Sender as TBaseRePort do
  begin
    SetFont('宋体',16);   //设置字体大小
    TBaseRePort(Sender).Canvas.Font.Color := clRed;//设置画布颜色
    Bold := True; //字体加粗
    UnderLine := True; //加下划线
    Ypos := 0.6; //设置打印高度
    PrintCenter('工资条',PageHeight/2); //居中打印
    SetFont('宋体',10);
    TBaseRePort(Sender).Canvas.Font.Color := clBlue;
    YPos := 0.9;
    PrintLeft('操作员:',0.4);
    PrintLeft(czymc,0.9);
    PrintLeft('时间:',PageHeight - 2);
    PrintLeft(FormatDateTime('yyyy-mm-dd',Now()),PageHeight - 1.4);
    Ypos := 1.2;
    ClearTabs; //清空当前表格打印方式
    SetFont('宋体',10);
    TBaseRePort(Sender).Canvas.Font.Color := clBlue;
    SetPen(clBlack,PsSolid,1,PmCopy);//设置画笔风格
    SetTab(0.4,pjCenter,0.9,0,BoxLineall,0);//设置新的表格打印方式，调用一次SetTab 表示
```

```
        //设置一个新的单元格
        tab(-2,na,-2,-2,10);//Tab用于跳转到下一个单元格
        Print('员工编号');
    end;
end;
```

SetTab 过程说明：

```
Procedure SetTab(NewPos: Double; NewJustify: TprintJustify; NewWidth: Double;NewMargin:
    Double; NewLines: Byte;NewShade: Byte);
```

NewPos 定义单元格的开始位置，如果为 Na，表示从前一个单元格结束的位置开始显示新的单元格。

NewJustify 表示单元格内文本的对齐方式。

NewWidth 表示表格宽度。

NewMargin 表示文本与表格线的距离。

NewLines 表示网格线布局。

NewShade 定义单元格的阴影浓度，用百分比表示。

Tab 过程说明：

```
Procedure Tab(LeftWidth: Integer; RightWidth: Integer; TopWidth: Integer; BottomWidth:
    Integer;ShadeOverride: Integer);
```

LeftWidth、RightWidth、TopWidth、BottomWidth 分别表示单元格左边框宽度、右边框宽度、上边框宽度、下边框宽度。

ShadeOverride 表示单元格阴影。

19.10.2 如何控制表格中焦点的移动

在使用表格录入数据时，为方便用户的录入，需要控制表格中焦点的移动。例如，用户在一个单元格中录入完数据，按 Enter 键，焦点将落在另一个单元格上。字符串网格控件 TStringGrid 提供了“Col”属性与“Row”属性，通过设置这两个属性便可以控制焦点的移动。程序代码如下：

```
if Key = VK_ReTurn then
begin
    if StringGrid1.Col < StringGrid1.ColCount-1 then
        StringGrid1.Col := StringGrid1.Col+1;
end;
```

19.10.3 编写简洁、灵活的代码

在开发程序时，代码要尽量编写得简洁、灵活，但前提是必须实现预定的功能。例如，在员工工资调整管理中，程序要判断每一个编辑框是否输入了合法的字符，如果在编辑框的 OnExit 事件中写如下代码，则需要在每一个编辑框中都写类似的代码。

```
procedure TForm84.Edit1Exit(Sender: TObject);
begin
    if Trim(Edit1.Text)<>'' then
        Try
```

```
        StrToFloat(Edit1.Text);
      Except
        Application.MessageBox('请输入合法字符。','提示',64);
        Edit1.Clear;
      end;
end;
```

如果将上面的代码稍做修改，让其他编辑框的 OnExit 事件调用这个通用的 OnExit 事件代码，便可以减少大量代码的编写。代码修改如下：

```
procedure TForm84.Edit1Exit(Sender: TObject);
begin
  if Trim(TEdit(Sender).Text)<>'' then
    Try
      StrToFloat(TEdit(Sender).Text);
    Except
      Application.MessageBox('请输入合法字符。','提示',64);
      TEdit(Sender).Clear;
    end;
end;
```

19.10.4 如何限制表格中某列的输入

在利用表格录入数据时，有时需要限制表格中数据的录入，例如，在金额、数量字段不允许输入字符。在程序中，可以定义两个全局变量，用来记录当前单元格的横、纵坐标。当纵坐标处于某一列时，便可以限制用户的录入。程序代码如下：

```
procedure TForm83.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
Begin
  //x、y是作者定义的两个全局整型变量，记录当前单元格的横、纵坐标
  y := ACol;
  x := ARow;
end;
//当当前单元格处于某一列时，限制用户输入
procedure TForm83.StringGrid1KeyPress(Sender: TObject; var Key: Char);
var
  mm: Boolean;
begin
  if(y = 6)or(y =8)then
  begin
    mm := (Key <#8)or(Key >#8)and(Key<#48)or(Key>#57);
    if mm then
      Key := #0;
  end
 end;
```

19.10.5 员工工资核算

员工工资核算总的思路是用员工各项收入工资减去员工各项应扣工资。例如，某一员

工的收入工资包括基本工资、职务工资、工龄工资、住房补助、生活补助、加班工资、上月余额，各项应扣工资包括养老保险金、个人所得税、保险费、事假扣款，那么，该员工应发工资为总的各项收入工资减去员工各项应扣工资。在工资发放时，如果员工应发工资与实发工资不一致，则用员工应发工资减去实发工资，作为本月余额。

19.10.6 查询上月余额

在进行职工工资发放时，需要查询该职工的上月余额，以计算本月应发工资。工资发放时，职工工资发放信息被存储在工资发放明细表中，因此，可以从工资发放明细表中查询上月余额。在查询上月余额时，需要注意，如果当前系统日期为 1 月份，则查询的时间条件为上一年的 12 月份。

19.11 软件测试

系统测试是对开发人员完成的软件进行检测，测试软件是否完成预期的功能，软件是否存在 Bug。有时，测试一个模块会比写一个模块花费的时间还要长，因为测试需要考虑方方面面，如果一个地方疏忽了，就会使软件存在漏洞，将来就可能会给用户带来损失。其实，一个不存在 Bug 的软件几乎是不存在的，例如，MicroSoft Windows 操作系统，据说测试 Windows 操作系统的工程师多达几百人，但是，Bug 依然存在，要不然也不会有那么多的补丁出现。下面以测试员工基本工资管理为例介绍测试软件的一般方法。

1. 测试软件是否完成预定的功能

（1）运行程序，打开基本工资管理窗口。效果如图 19.15 所示。



图 19.15 基本工资管理窗口

（2）在表格中输入基本工资、职务工资、工龄工资、养老保险金，效果如图 19.16 所示。

（3）单击“保存”按钮，如果程序出现如图 19.17 所示的提示，表明保存成功。

2. 测试软件是否健壮

一个好的软件除了能够完成用户的基本操作，还应该具有良好的健壮性，即具有很强的错误处理机制和防错误功能。对于一个软件，用户按正规操作，可能不会出现问题，如果用户不小心进行了误操作，可能会导致难以预料的后果，这就要求软件应具有错误处理机制和防错误

功能。例如，用户在设置员工基本工资时，如果不是输入类似“1900”的数字，而输入“abc”字符串。这样的操作显然是一个违规操作，但是，对于优秀的软件，应该阻止此类错误的出现。



图 19.16　员工基本工资管理



图 19.17　提示信息

19.12　程序设计清单

工资管理系统文件清单如表 19.14 所示。

表 19.14　　工资管理设计清单

文　件　名	文 件 类 型	说　　明
Dataa.dfm	窗体文件	数据模块
Gzffglxt.dfm	窗体文件	工资发放管理窗体
Gzglxt.dfm	窗体文件	主窗体
Gzmxbb.dfm	窗体文件	工资明细报表窗体
Gztzglxt.dfm	窗体文件	工资调整窗体
Mp.dfm	窗体文件	模板窗体
Rysc.dfm	窗体文件	员工信息册除窗体
Rytj.dfm	窗体文件	员工信息添加窗体
Tjyusc.dfm	窗体文件	操作员管理窗体
Xtdl3.dfm	窗体文件	系统登录窗体
Ygfdgzgl.dfm	窗体文件	员工浮动工资管理窗体
Ygjbgzgl.dfm	窗体文件	员工基本工资管理窗体
Gzgl.dpr	工程文件	
Gzgl.exe	可执行文件	

19.13　技术、经验总结

19.13.1　创建视图

在本系统的员工工资调整管理模块中，作者是通过修改视图来间接修改员工基本工资

表、员工浮动工资项目表数据的。视图是执行“Select”语句所得到的查询结果集。通过修改结果集中的数据，可以间接修改基表中的数据。下面介绍如何创建视图。

（1）打开企业管理器，展开“数据库”选项，选择数据库，由于本系统使用的数据库为“工资管理系统”，因此，本节实例选择“工资管理系统”数据库。

（2）展开“工资管理系统”选项，用鼠标右键单击“视图”选项，将弹出一个快捷菜单，如图 19.18 所示。

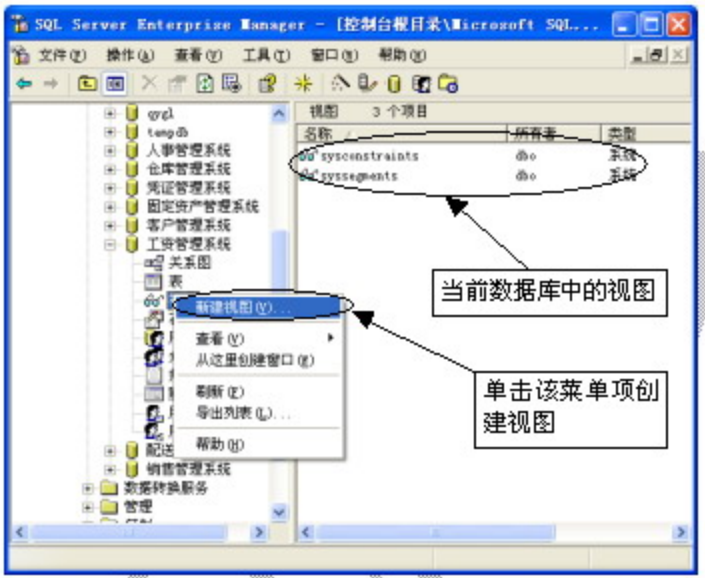


图 19.18　创建视图

（3）单击“新建视图”菜单项，打开如图 19.19 所示的窗口。



图 19.19　新建视图

（4）在代码段中输入如下代码：

```
SELECT 员工基本工资表.*, 员工浮动工资项目表.个人所得税 AS 个人所得税,
       员工浮动工资项目表.住房补助 AS 住房补助,
       员工浮动工资项目表.保险费 AS 保险费,
       员工浮动工资项目表.生活补助 AS 生活补助
FROM 员工基本工资表 ,
```

```
        员工浮动工资项目表
where
        员工基本工资表.员工编号 = 员工浮动工资项目表.员工编号
```

(5) 单击“运行”按钮，执行 SQL 语句。

(6) 单击“保存”按钮，保存视图为“工资发放”。

19.13.2 经验总结

如果一个窗体是从另一个父窗体派生的，那么在子窗体中是不允许删除父窗体中的组件的。例如，在一个父窗体中放置一个 TPanel 组件，命名为“Panel1”，在子窗体中删除“Panel1”时，会出现错误提示，即当前选中的组件是在祖先类中引入的，不允许删除。错误提示如图 19.20 所示。



图 19.20 错误提示

明日科技

第 20 章 Delphi 在人事管理系统中的应用

20.1 人事管理系统概述

20.1.1 开发背景及意义

人事管理是各企事业、行政、科研单位所必需的，它需要对人事档案进行保存、统计、更新。如果人工管理，工作量会很大，尤其是在比较大的企业。当企业职工达到上万人，人工管理似乎不可想象，例如，从一万条职工信息中查找一条职工信息，手工查找会很费时、费力，如果需要统计职工信息，更是难上加难。

随着社会信息化的发展和计算机的普及，以上问题便迎刃而解了，计算机的使用使人们从繁重的劳动中解脱出来，即使是从几百万条信息中统计、查找数据，在几分钟之内便可以完成。

20.1.2 系统目标

- 实现公司、部门信息管理。
- 实现员工信息管理，包括员工信息的添加、修改、删除。
- 实现员工调动管理，员工可以被调到任何部门。
- 实现系统登录管理，提高系统安全性，防止非法人员查看人事档案。

20.2 系统分析

20.2.1 系统规划

根据上面的系统功能描述，可以定义如下需求分析。

1. 系统目标

利用计算机系统保存人事档案，简化人工操作流程，提高信息安全性。

2. 系统范围

- 基础信息管理
- 人员管理

3. 系统功能

- 公司信息维护
- 部门信息管理

在部门信息管理中需要实现部门信息的浏览、添加、修改、删除。

- 人员类别信息管理
- 银行信息管理
- 员工信息管理

在员工信息管理中需要实现员工档案的添加、修改、删除。

- 员工调动管理
- 操作员管理

在操作员管理中需要实现操作员信息的添加、删除及权限设置。

- 系统结构图

人事管理系统的结构图如图 20.1 所示。

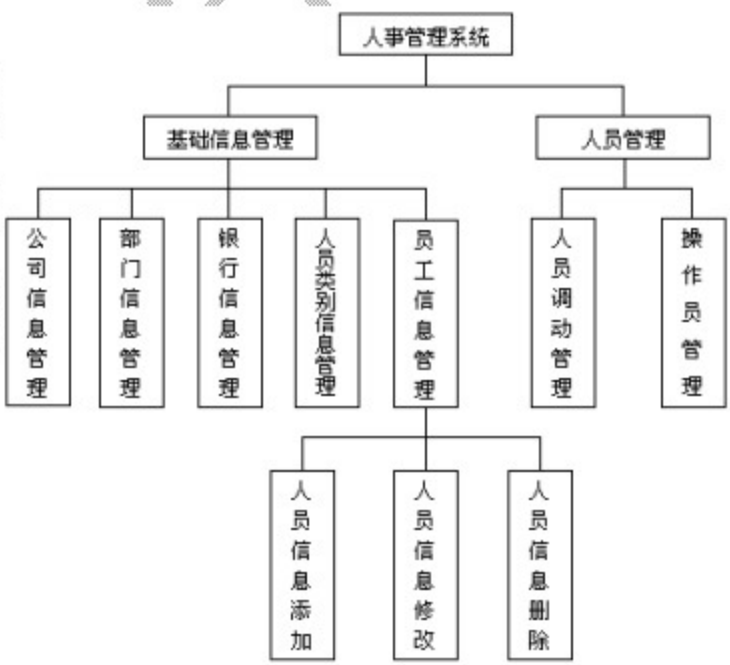


图 20.1　人事管理系统结构图

20.2.2 系统流程图

人事管理系统的流程图如图 20.2 所示。

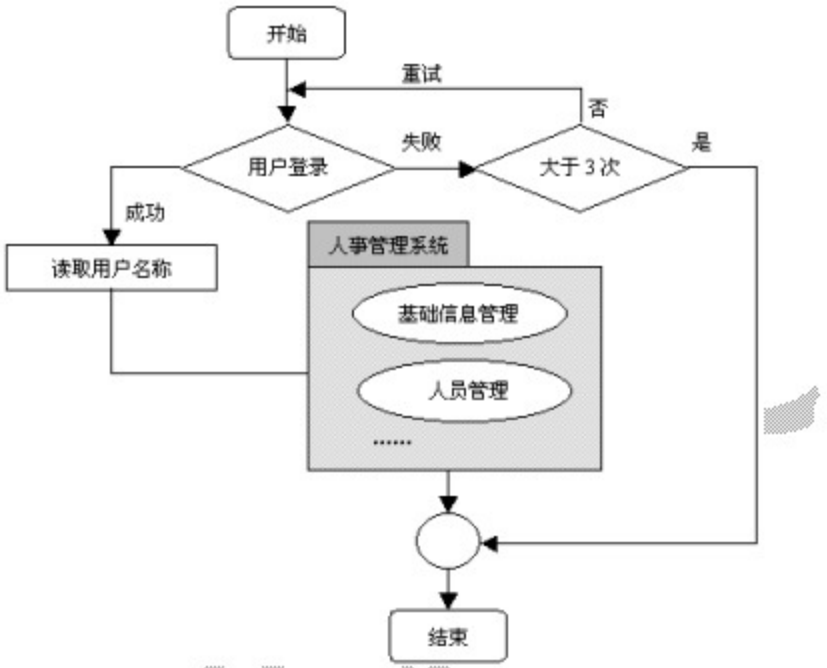


图 20.2 人事管理系统流程图

20.3 数据库设计

20.3.1 数据库概念设计

人事管理系统的 E-R 图如图 20.3 所示。

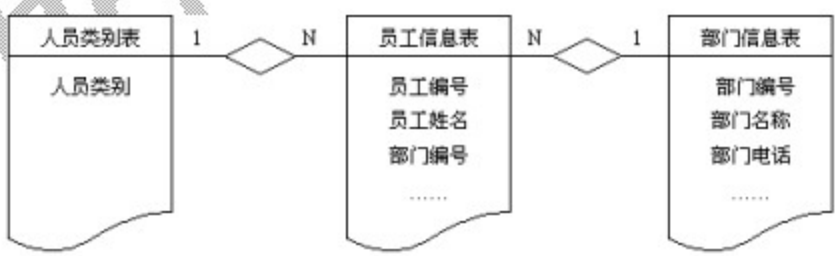


图 20.3 人事管理系统 E-R 图

20.3.2 数据库逻辑结构设计

人事管理系统共使用了 6 张数据表，分别为：操作员信息表、人员类别表、员工基础信息表、银行信息表、部门信息表、公司信息表。由于篇幅关系，在此只列出使用较多的数据表结构（如表 20.1 和表 20.2 所示），其他表结构请参考光盘。

表 20.1　　员工基础信息表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
员工编号	Varchar(30)	是		否	
员工姓名	Varchar(20)				
性别	Varchar(20)				
文化程度	Varchar(20)				
部门编号	Varchar(30)				
部门名称	Varchar(30)				
人员类别	Varchar(30)				
聘用日期	DateTime				GetDate()
停发工资	Int				
计税	Int				
中方人员	Int				
离职	Int				
代发银行	Varchar(30)				
银行账号	Varchar(30)				
备注	Varchar(60)				

表 20.2　　部门信息表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
部门编号	Varchar(30)	是		否	
部门名称	Varchar(20)				
部门电话	Varchar(20)				
备注	Varchar(20)				

20.4　处理过程分析

20.4.1　人员档案处理过程分析

为了实现程序结构的模块化，可以将人员档案管理按功能分成 3 个子模块，在主模块中单击相应的按钮可以调出子模块。在添加人员信息子模块中，添加人员信息时，首先需要进行登记，由程序产生员工编号，然后录入完员工信息后，单击“保存”按钮便可以将员工信息保存到数据库中。在修改人员信息子模块中，首先选择查询条件将人员信息查询出来，然后修改人员信息，将修改后的信息保存到数据库中。在删除人员信息子模块中，首先将欲删除的人员信息查询出来，然后删除人员信息。人员档案业务流程如图 20.4 所示。

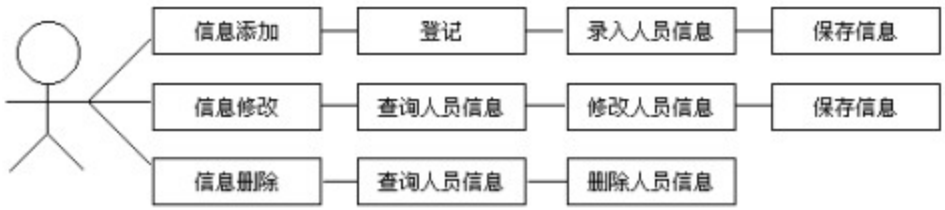


图 20.4　人员档案业务流程图

20.4.2 人员调动处理过程分析

要实现员工在部门间的调动，首先需要将员工信息（包括员工部门信息）查询出来，然后修改部门信息，最后将修改后的信息保存到数据表中。在查询时，为辅助用户设置查询条件，当用户选择一个字段时，在另一个组合框中会显示相应的字段值。人员调动业务流程如图 20.5 所示。

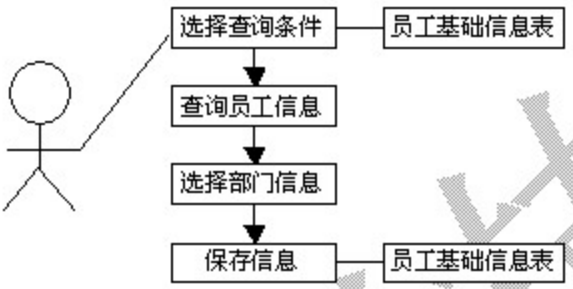


图 20.5 人员调动流程图

20.5 主窗体设计

主窗体设计步骤如下。

（1）创建一个窗体，在窗体上放置 TImage、TSpeedButton、TLabel 等组件，如图 20.6 所示。



图 20.6 主窗体界面

（2）设置各主要组件属性如表 20.3 所示。

表 20.3 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form74	TForm	BorderStyle	bsNone
		Font.Size	8
SpeedButton1	TSpeedButton	Flat	True
		Cursor	crHandPoint

20.6 数据模块设计

数据模块设计步骤如下。

（1）启动 Dephi 7，打开工程“rsgl”，单击“File”/“New”/“Data Module”菜单项，创建数据模块。

（2）在数据模块上放置 TADOConnection 组件、TADOQuery 组件，如图 20.7 所示。



图 20.7 数据模块

（3）设置各主要组件属性如表 20.4 所示。

表 20.4 组件属性设置

对 象 名	对 象 类 型	属 性	值
ADOConnection1	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=人事管理系统
ADOQuery1	TADOQuery	Connection	ADOConnection1
ADOQuery2	TADOQuery	Connection	ADOConnection1
ADOQuery3	TADOQuery	Connection	ADOConnection1

20.7 程序设计与编码

20.7.1 员工信息管理设计

1. 概述

员工信息管理是人事管理系统的核心。员工信息管理类似于一个主窗口，窗体显示时，在表格中会显示当前所有的员工信息。而且，通过功能按钮，能够调出员工信息添加、修改、删除窗口。员工信息管理运行结果如图 20.8 所示。

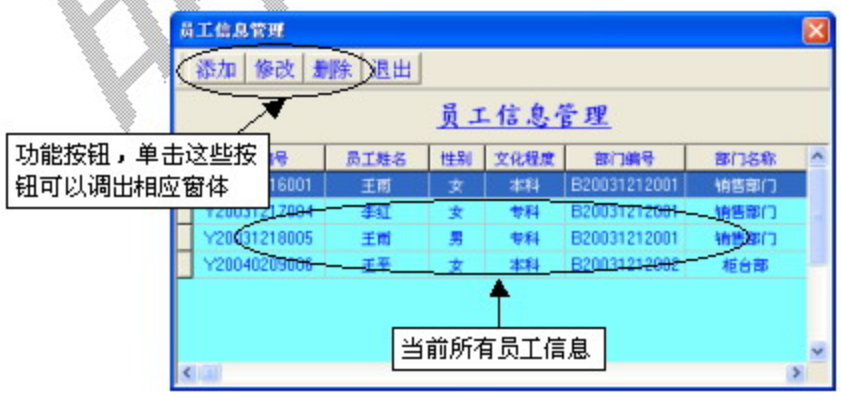


图 20.8 员工信息管理

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TSpeedButton、TLabel、TDataSource、TDBGrid、TToolBar、TCoolBar 等组件。

（2）设置各主要组件属性如表 20.5 所示。

表 20.5　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form11	TForm	Font.Color	clBlue
		Font.Size	10
		BorderStyle	bsDialog
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

在窗体显示时，将员工基础信息表中的数据显示在列表中，即加载已有的员工信息。

```
procedure TForm11.FormShow(Sender: TObject);
begin
  with Data3.ADOQuery5 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 员工基础信息表');
    Open;
  end;
  if Data3.ADOQuery5.RecordCount>0 then
    DataSource1.DataSet := Data3.ADOQuery5;
end;
```

下面是“删除”按钮的事件代码，单击“删除”按钮会调用删除人员档案管理窗体。

```
procedure TForm11.SpeedButton1Click(Sender: TObject);
begin
  Application.CreateForm(TForm79,Form79);
  Form79.ShowModal;
  Form79.Free;
end;
```

下面是“添加”按钮的单击事件代码，单击“添加”按钮会调用添加人员档案管理窗体。

```
procedure TForm11.SpeedButton2Click(Sender: TObject);
begin
  Application.CreateForm(TForm77,Form77);
  Form77.ShowModal;
  Form77.Free;
end;
```

20.7.2　添加人员档案程序设计

1. 概述

添加人员档案是将企业员工信息保存到数据库中。在添加员工信息时，系统会自动生

成员工编号，当操作员录入完员工信息后，单击“保存”按钮会将用户信息保存起来，如果想放弃已录入的数据，单击“取消”按钮可以清除已录入的数据。添加人员档案运行结果如图 20.9 所示。

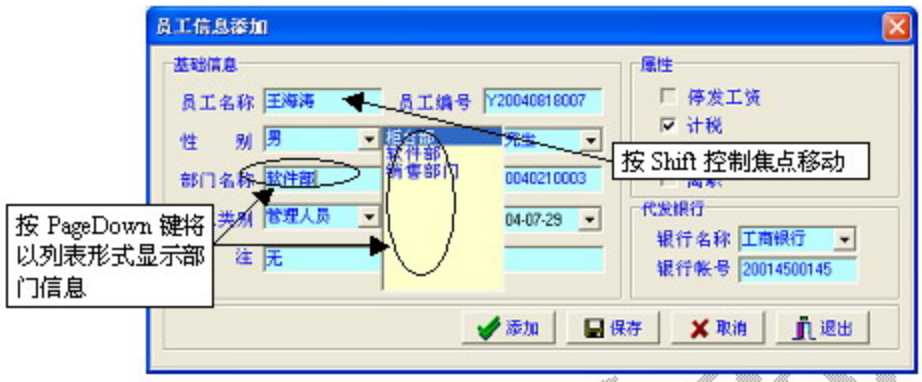


图 20.9　添加人员档案

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TDataSource、TDBGrid、TCheckBox、TGroupBox 等组件。

（2）设置各主要组件属性如表 20.6 所示。

表 20.6　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form77	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
DBGrid1	TDBGrid	DataSource	DataSource1
BitBtn2	TBitBtn	Enabled	False

3. 程序相关代码

下面是“保存”按钮的单击事件代码，程序会将员工信息保存到数据表中。在保存员工信息时，首先检查该员工是否存在，如果存在，进行提示并停止操作，否则利用 Insert 语句添加数据。

```
procedure TForm77.BitBtn2Click(Sender: TObject);
begin
  inherited;
  With Data3.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 员工基础信息表 where 员工编号 = :a');
    Parameters.ParamByName('a').Value := Trim(Edit1.Text);
    Open;
  end;
```

```
    if Data3.ADOQuery1.RecordCount > 0 then
    begin
      Application.MessageBox('该员工已经存在。','提示',0+64);
      Exit;
    end;
    Try
    With Data3.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      //添加SQL语句
      SQL.Add('Insert 员工基础信息表 values (
        :a,:b,:c,:d,:e,:f,:h,default,:j,:k,:l,:m,:n,:o,:p)');
      Parameters.ParamByName('a').Value := Trim(Edit1.Text);
      Parameters.ParamByName('b').Value := Trim(Edit2.Text);
      Parameters.ParamByName('c').Value := Trim(ComboBox1.Text);
      Parameters.ParamByName('d').Value := Trim(ComboBox2.Text);
      Parameters.ParamByName('e').Value := Trim(Edit4.Text);
      Parameters.ParamByName('f').Value := Trim(Edit3.Text);
      Parameters.ParamByName('h').Value := Trim(ComboBox4.Text);
      //零表示没有停发工资，由于是添加操作员，所以该选项不可用
      Parameters.ParamByName('j').Value := 0;
      //0表示计税，1表示不计税
      if CheckBox2.Checked = True then
        Parameters.ParamByName('k').Value := 0
      else
        Parameters.ParamByName('k').Value := 1;
      if CheckBox3.Checked = True then
        Parameters.ParamByName('l').Value := 0
      else
        Parameters.ParamByName('l').Value := 1;
      Parameters.ParamByName('m').Value := 0;
      Parameters.ParamByName('n').Value := Trim(ComboBox3.Text);
      Parameters.ParamByName('o').Value := Trim(Edit5.Text);
      Parameters.ParamByName('p').Value := Edit7.Text;
      ExecSQL;
    end;
    Application.MessageBox('保存成功。','提示',0+64);
    BitBtn3.OnClick(Sender);
    Except
    Application.MessageBox('系统出错。','提示',0+64);
    Close;
    end;
  end;
```

自定义过程 FindNextCtrl 控制焦点移动。用户在录入信息时，按 Enter 键将使焦点移到另一个控件上。FindNextCtrl 过程调用了 FindnextControl 方法，该方法用于查找下一个控件，共有 3 个参数，第一个参数用于确定当前控件；第二个参数控制查找方向，为 True，

向前查找，为False，向后查找；第三个参数、第四个参数确定方法是否检查发现的控件，如果第三个参数为True，返回的控件TabStop属性必须设置为True，如果第四个参数为True，窗体中的直系子控件将获得焦点。

```
procedure TForm77.FindNextCtrl;
begin
  if FindnextControl(activeControl,True,true,False)is TEdit then
    TEdit(FindnextControl(activeControl,True,true,False)).SetFocus
  else if  FindnextControl(activeControl,True,true,False)is TComboBox then
    TComboBox(FindnextControl(activeControl,True,true,False)).SetFocus
  else if FindnextControl(activeControl,True,true,False)is TDateTimePicker then
    TDateTimePicker(FindnextControl(activeControl,True,true,False)).SetFocus;
end;
```

20.7.3 修改人员档案程序设计

1. 概述

当企业的人员信息有变动时，操作员需要修改员工信息。修改人员档案就是为了适应企业员工信息不断变化而设计的。在修改人员信息时，首先需要查询欲修改的人员信息，将其显示在界面中，在用户修改人员信息后，单击“修改”按钮，可将员工信息保存起来。修改人员档案运行结果如图20.10所示。



图20.10 员工信息修改

2. 设计步骤

（1）新建一个窗体，在窗体上放置TPanel、TBitBtn、TSpeedButton、TLabel、TEdit、TComboBox、TDataSource、TDBGrid、TCheckBox等组件。

（2）设置各主要组件属性如表20.7所示。

表20.7 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form78	TForm	Font.Color	clBlue
		Font.Size	10
		BorderStyle	bsDialog
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

窗体显示时向组合框中输入人员类别和银行名称，用户在修改人员档案时，只要在组合框中选择信息即可，省去了手工输入的麻烦。

```
procedure TForm78.FormShow(Sender: TObject);
begin
  inherited;
  //查询员工信息表
  with Data3.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 员工基础信息表');
    Open;
  end;
  if Data3.ADOQuery1.RecordCount<1 then
  begin
    Application.MessageBox('没有人员信息可以修改。','提示',64);
    BitBtn1.Enabled := False;
    BitBtn2.Enabled := False;
    BitBtn3.Enabled := False;
    BitBtn4.Enabled := False;
    BitBtn5.Enabled := False;
    Exit;
  end;
  Clear;
  ComboBox2.Clear;
  ComboBox4.Items.Clear;
  with Data3.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 人员类别表');
    Open;
  end;
  while Not Data3.ADOQuery1.Eof do //利用循环向组合框中添加数据
  begin
    ComboBox4.Items.Add(Data3.ADOQuery1.FieldByName('人员类别').Value);
    Data3.ADOQuery1.Next;
  end;
  ComboBox5.Items.Clear;
  with Data3.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 银行信息表');
```

```
    Open;
  end;
  while Not Data3.ADOQuery1.Eof do
  begin
    ComboBox5.Items.Add(Data3.ADOQuery1.FieldByName('银行名称').Value);
    Data3.ADOQuery1.Next;
  end;
end;
```

下面是“查询”按钮的事件代码，程序会根据用户选择的条件返回相应的结果集。在查询时，首先判断是否显示全部信息（CheckBox5 是否选中），如果显示全部信息，不考虑是否设置了查询条件，直接从员工基础信息表中查询所有数据，否则（CheckBox5 没有选中）判断用户是否设置了查询条件，即“ComboBox1”或“ComboBox2”是否为空，如果为空，进行提示，并退出查询，否则根据查询条件从员工基础信息表中查询数据。

```
procedure TForm78.SpeedButton1Click(Sender: TObject);
begin
  if CheckBox5.Checked = False then //是否显示全部信息
  begin
    if (Trim(ComboBox1.Text)<>'')and(Trim(ComboBox2.Text)<>'')then //是否设置查询条件
    begin
      with Data3.ADOQuery2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 员工基础信息表 where');
        SQL.Add(Trim(ComboBox1.Text));
        SQL.Add('= ');
        SQL.Add(':a');
        Parameters.ParamByName('a').Value := Trim(ComboBox2.Text);
        Open;
      end;
      if Data3.ADOQuery2.RecordCount<1 then
        Clear
      else
      begin
        DataSource1.DataSet := Data3.ADOQuery2;
        EditValue;
      end;
    end
  end
  else //显示全部信息
  begin
    with Data3.ADOQuery2 do
    begin
      Close;
```

```
        SQL.Clear;
        SQL.Add('select * from 员工基础信息表 ');
       Open;
      end;
      if Data3.ADOQuery2.RecordCount<1 then
        Clear
      else
      begin
        DataSource1.DataSet := Data3.ADOQuery2;
        EditValue;
      end;
    end;
end;
```

自定义过程 EditValue 会将数据集当前记录数据显示在界面中。当用户在表格中选中一行数据时，该数据会显示在界面中。

```
procedure TForm78.EditValue;
begin
  if Data3.ADOQuery2.RecordCount>0 then
  begin
    Label6.Caption := Data3.ADOQuery2.FieldByName('员工编号').Value;
    Label9.Caption := Data3.ADOQuery2.FieldByName('员工姓名').Value;
    ComboBox3.Text := Data3.ADOQuery2.FieldByName('文化程度').Value;
    ComboBox4.Text := Data3.ADOQuery2.FieldByName('人员类别').Value;
    ComboBox5.Text := Data3.ADOQuery2.FieldByName('代发银行').Value;
    Edit2.Text := Data3.ADOQuery2.FieldByName('银行账号').Value;
    if Data3.ADOQuery2.FieldByName('停发工资').Value = 0 then
       CheckBox1.Checked := False
    else
      CheckBox1.Checked := True;
    if Data3.ADOQuery2.FieldByName('计税').Value = 0 then
       CheckBox2.Checked := True
    else
      CheckBox2.Checked := False;
    if Data3.ADOQuery2.FieldByName('中方人员').Value = 0 then
       CheckBox3.Checked := True
    else
      CheckBox3.Checked := False;
    if Data3.ADOQuery2.FieldByName('离职').Value = 0 then
       CheckBox4.Checked := False
    else
      CheckBox4.Checked := True;
  end;
end;
```

以下是“修改”按钮的单击事件代码，在修改数据时，程序首先提示用户是否真的修改数据，防止意外更改数据，然后检查现有数据是否为空，如果为空进行提示，并退出修

改操作，否则执行修改操作。

```
procedure TForm78.BitBtn1Click(Sender: TObject);
begin
  inherited;
  if Data3.ADOQuery2.RecordCount>0 then
  begin
    if Application.MessageBox('确实要修改当前记录吗？','提示',MB_YesNo)= ID_Yes then
    begin
      if (Trim(ComboBox3.Text)<>'')and(Trim(ComboBox4.Text)<>'')and
        (Trim(ComboBox5.Text)<>'')and(Trim(Edit2.Text)<>'') then
      begin
        Try
          Data3.ADOQuery2.Edit;//使数据集处于编辑状态
          Data3.ADOQuery2.FieldByName('文化程度').Value := Trim(ComboBox3.Text);
          Data3.ADOQuery2.FieldByName('人员类别').Value := Trim(ComboBox4.Text);
          Data3.ADOQuery2.FieldByName('代发银行').Value := Trim(ComboBox5.Text);
          Data3.ADOQuery2.FieldByName('银行账号').Value := Trim(Edit2.Text);
          if CheckBox1.Checked = True then
            Data3.ADOQuery2.FieldByName('停发工资').Value := 1
          else
            Data3.ADOQuery2.FieldByName('停发工资').Value := 0;
          if CheckBox2.Checked = True then
            Data3.ADOQuery2.FieldByName('计税').Value := 0
          else
            Data3.ADOQuery2.FieldByName('计税').Value := 1;
          if CheckBox3.Checked = True then
            Data3.ADOQuery2.FieldByName('中方人员').Value := 0
          else
            Data3.ADOQuery2.FieldByName('中方人员').Value := 1;
          if CheckBox4.Checked = True then
            Data3.ADOQuery2.FieldByName('离职').Value := 1
          else
            Data3.ADOQuery2.FieldByName('离职').Value := 0;
          Data3.ADOQuery2.Post;//提交对数据集所做的修改
          Application.MessageBox('数据修改成功。','提示',64);
        Except
          Application.MessageBox('系统出错。','提示',64);
          Close;
        end;
      end
      else
        Application.MessageBox('数据项不能为空.','提示',64);
    end
    else
      EditValue;
  end;
end;
```

20.7.4 删除人员档案管理设计

1. 概述

删除人员档案是将企业不需要的员工信息删除，以释放更多的数据库空间。删除人员档案与修改人员档案操作步骤大致相同，首先需要查询欲删除的人员信息，将其显示在界面上，操作员在删除数据时，程序会提示是否真的要删除人员信息，单击“是”按钮会将员工信息删除，单击“否”按钮取消删除操作。删除人员档案运行结果如图 20.11 所示。

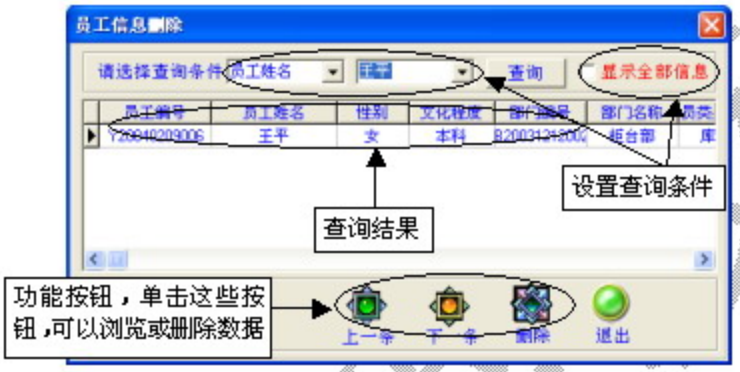


图 20.11 员工信息删除

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TSpeedButton、TLabel、TCheckBox、TDataSource、TDBGrid，TImage 等组件。

（2）设置各主要组件属性如表 20.8 所示。

表 20.8 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form79	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
CheckBox1	TCheckBox	Checked	False
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

下面是“查询”按钮的事件代码，程序会根据用户选择的条件返回相应的结果集。在查询时，首先判断是否显示全部信息（CheckBox5 是否选中），如果显示全部信息，不考虑是否设置了查询条件，直接从员工基础信息表中查询所有数据，否则（CheckBox5 没有选中）判断用户是否设置了查询条件，即“ComboBox1”或“ComboBox2”是否为空，如果为空，进行提示，并退出查询，否则根据查询条件从员工基础信息表中查询数据。

```
procedure TForm79.SpeedButton1Click(Sender: TObject);
begin
  inherited;
  if CheckBox5.Checked = False then
```

```
begin
  if (Trim(ComboBox1.Text)<>'')and(Trim(ComboBox2.Text)<>'')then
  begin
    with Data3.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 员工基础信息表 where');
      SQL.Add(Trim(ComboBox1.Text));
      SQL.Add('= ');
      SQL.Add(':a');
      Parameters.ParamByName('a').Value := Trim(ComboBox2.Text);
      Open;
    end;
    if Data3.ADOQuery2.RecordCount>0 then
      DataSource1.DataSet := Data3.ADOQuery2;
  end
  else
  begin
    with Data3.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 员工基础信息表 ');
     Open;
    end;
    if Data3.ADOQuery2.RecordCount>0 then
      DataSource1.DataSet := Data3.ADOQuery2;
  end;
end;
```

在删除人员档案时，首先判断当前记录集是否有数据，如果存在可删除的数据，将提示是否进行删除操作，在提示信息框中单击“是”按钮将执行删除操作。

```
procedure TForm79.Image4Click(Sender: TObject);
begin
  inherited;
  if Data3.ADOQuery2.Active = True then
    if Application.MessageBox('确实要删除当前记录吗？','提示',MB_YesNo)= ID_Yes then
      Data3.ADOQuery2.Delete;
end;
```

20.7.5 员工调动管理设计

1. 概述

员工调动管理是将企业员工从一个部门调到另一个部门，或从一个车间调到另一个车

间。为了使企业的人力资源能够得到合理的利用，企业经常需要调整人员，员工调动管理正是为企业调整人力资源而设计的。员工调动管理运行结果如图 20.12 所示。



图 20.12　员工调动管理

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TSpeedButton、TLabel、TComboBox、TDataSource、TDBGrid，TImage、TListBox 等组件。

（2）设置各主要组件属性如表 20.9 所示。

表 20.9　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form79	TForm	Font.Color	clBlue
		Font.Size	8
ListBox1	TlistBox	Viseble	False
DBGrid1	TDBGrid	DataSource	DataSource1
		Visible	False

3. 程序相关代码

处理 ComboBox1 的 OnChange 事件，当 ComboBox1 内容变化时，ComboBox2 中的内容会跟着相应地改变，方便用户设置查询条件。

```
procedure TForm80.ComboBox1Change(Sender: TObject);
begin
  inherited;
  if Trim(ComboBox1.Text)<>'' then
  begin
    with Data3.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select distinct');
      SQL.Add(Trim(ComboBox1.Text));
      SQL.Add('From 员工基础信息表');
      Open;
    end;
    if Data3.ADOQuery1.RecordCount>0 then
```

```
    begin
      ComboBox2.Clear;
      while Not Data3.ADOQuery1.Eof do
      begin
        ComboBox2.Items.Add(Data3.ADOQuery1.Fields[0].Value);
        Data3.ADOQuery1.Next;
      end;
    end;
  end;
end;
```

以下是“修改”按钮的单击事件。程序首先判断基础信息是否为空，如果为空，退出修改操作，反之，提示是否进行修改操作，单击“是”按钮将修改员工的部门信息。

```
procedure TForm80.Image3Click(Sender: TObject);
begin
  inherited;
  if (Trim(Label2.Caption)<>'')and(Trim(Label7.Caption)<>'')and
  (Trim(Label9.Caption)<>'')and(Trim(ComboBox3.Text)<>'')and(Trim(Edit1.Text)<>'') then
    if Application.MessageBox('确实要修改该员工部门信息吗？','提示',MB_YesNo)= ID_Yes then
    begin
      Try
        with Data3.ADOQuery2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('Update 员工基础信息表 set 部门名称 = :a, 部门编号 = :b
           where 员工编号 = :c');
          Parameters.ParamByName('a').Value := Trim(ComboBox3.Text);
          Parameters.ParamByName('b').Value := Trim(Edit1.Text);
          Parameters.ParamByName('c').Value := Trim(Label2.Caption);
          ExecSQL;
        end;
        Clear;
        Application.MessageBox('数据修改成功。','提示',64);
      Except
        Application.MessageBox('系统出错。','提示',64);
        Close;
      end;
    end;
end;
```

20.8 疑难问题分析解决

20.8.1 如何将数据表中的字段值添加到下拉列表框中

TComboBox 组件为组合框，结合了文本框和列表框的功能，使用户可以很方便地从列

表框中选择所需的列表项。所以在一些数据录入或查询中经常使用 TComboBox 组件提高录入或查询效率。在使用 TComboBox 组件作为数据选择时需要将选择字段赋值给 TComboBox 组件。在设计时可以通过 Items 属性手工添加数据表字段到 TComboBox 组件，这种方法的好处是不需编写代码，缺点一是录入数据浪费了时间，也可能录入错误数据；二是不灵活，一旦数据表改变或数据表字段改变，需要重新修改 TComboBox 组件中的数据。使用代码的方法解决了上述两个问题。下面介绍两种方法将数据库字段内容添加到 TComboBox 组件。

- 在设计时向组合框中添加数据

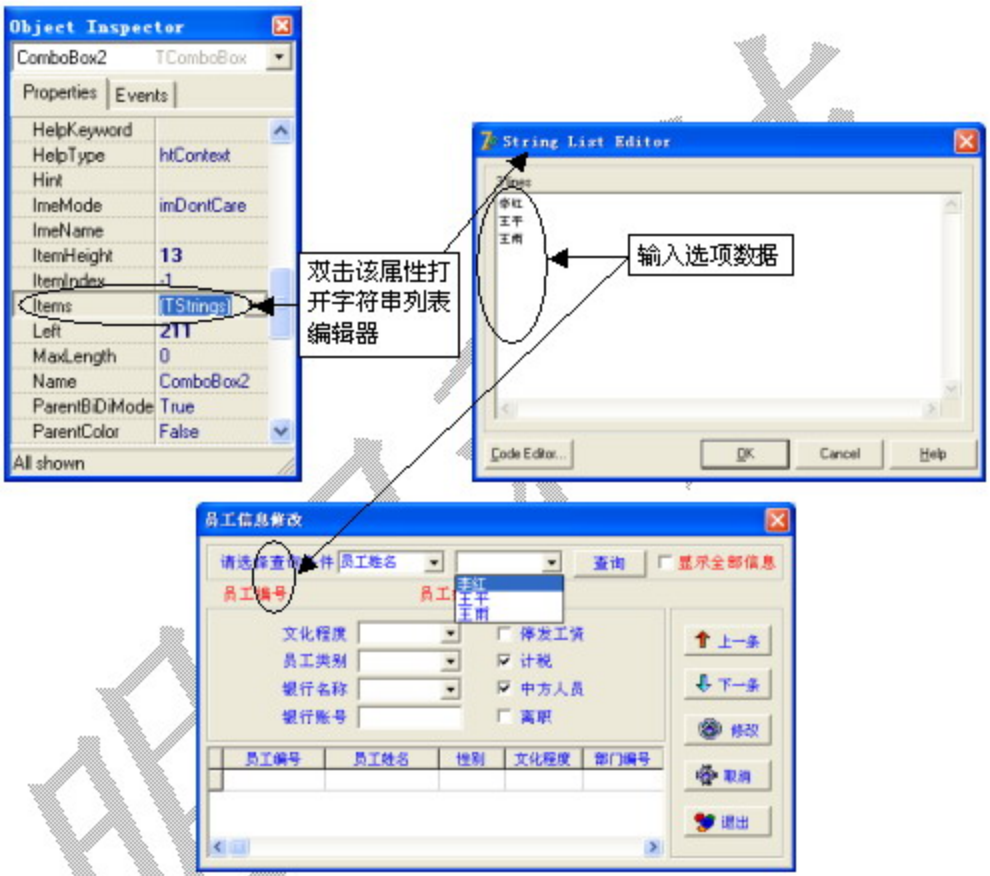


图 20.13　向组合框中添加数据

- 编写代码将数据添加到组合框中

```
ComboBox1.Items.Add('李红');
ComboBox1.Items.Add('王平');
```

将数据表中某一字段值添加到组合框中，代码如下：

```
with Data3.ADOQuery1 do
  begin
    Close; //关闭数据集
    SQL.Clear;//清空SQL语句
    SQL.Add('select * from 人员类别表'); //添加SQL语句
    Open; //打开数据集
  end;
```

```
  ComboBox4.Items.Clear; //清空下拉列表框中的数据
//当从数据库中查询数据时，默认情况下，当前指针会在结果集的第一条记录上，所以不需要写
// Data3.ADOQuery1.First 语句
while Not Data3.ADOQuery1.Eof do //使用循环将字段值添加到下拉列表框中
 begin
    //首先将第一行记录中的字段值添加到下拉列表框中
    ComboBox4.Items.Add(Data3.ADOQuery1.FieldByName('人员类别').Value);
    Data3.ADOQuery1.Next;//改变当前记录，使下一行记录成为当前记录
  end;
```

在上面的循环语句中，一定要写“Data3.ADOQuery1.Next”语句，如果不写该语句，循环将没有结束条件，程序会陷入死循环。

用户在使用组合框控件 TComboBox 时，如果想设置组合框当前文本的内容，可以有两种方法。

（1）设置 Text 属性

通过 Text 属性，用户可以读取或修改组合框中当前的文本。设置 Text 属性，既可以通过对象观察器设置，也可以通过程序代码设置。

- 使用对象观察器设置 Text 属性。具体如图 20.14 所示。

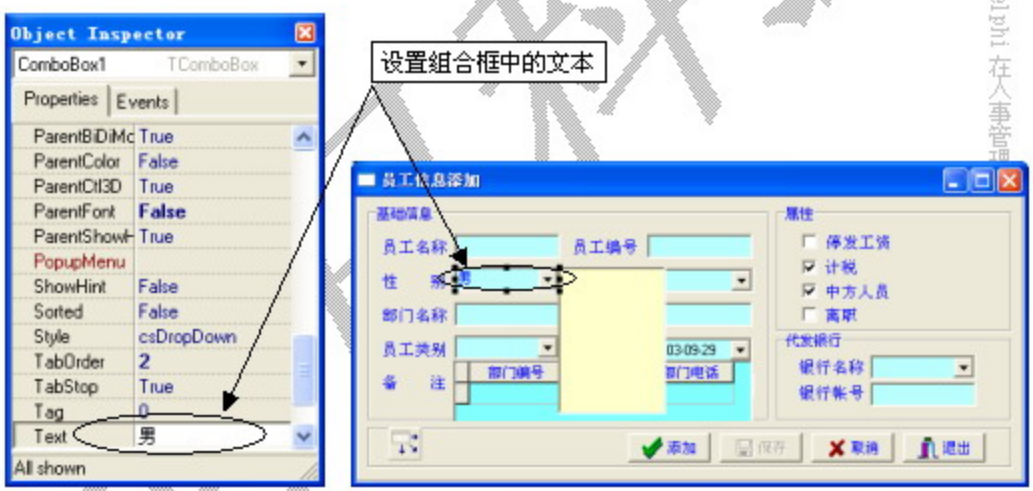


图 20.14 设置组合框文本

- 使用代码设置组合框文本

```
ComboBox1.Text := '男';
```

（2）设置 ItemIndex 属性

通过设置组合框的 ItemIndex 属性，可以将组合框中的某一选项设置为当前内容。组合框中的各选项是按索引顺序排列的，其索引是从 0 开始的，即组合框中第一个选项的索引为 0，第二个选项的索引为 1，依次类推。下面的程序代码将组合框中第一个选项设置为当前文本。

```
ComboBox1.ItemIndex := 0;
```

20.8.2 如何防止用户改变下拉列表框中的当前数据

在编写程序时，为了方便用户录入数据，经常将一些可选数据添加到下拉列表框中，让用

户选择，例如，在“性别”选项中，让用户选择男、女。但是，在 Delphi 中，用户可以修改下拉列表框中的当前数据。如何防止用户改变下拉列表框中的当前数据呢？请看下面的程序代码：

```
procedure TForm78.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
    Key := #0;
end;
```

当用户在下拉列表框中按键时，会触发下拉列表框的 OnKeyPress 事件，在该事件中，将用户所按键的键值置为空，这样，用户在下拉列表框中就不会输入任何数据。

20.8.3 隐藏窗体的标题栏

在本章的人事管理系统中，在主窗体运行时只显示一幅画面，而没有标题栏。如图 20.15 所示。



图 20.15 人事管理系统主窗口

在 Delphi 中，隐藏窗体的标题栏非常简单，只需要将“BorderStyle”属性设置为“bsNone”即可。但是，在窗体中需要提供一个关闭窗口的接口，否则用户无法关闭该窗口。

20.8.4 保存的实现方法探究

在开发数据库应用程序时，经常需要保存数据。例如，基本信息的录入、修改等。在保存数据时，需要保证用户录入数据的完整性。为了防止用户信息的残缺，可以在保存时检查用户信息是否完整，如果不完整，不允许保存，同时进行提示。在本章程序的人员添加管理模块中，作者采用了另一种方法，首先使“保存”按钮不可用，在用户输入数据时，检查数据是否完整，如果完整，“保存”按钮可用，否则不可用。代码如下：

```
procedure TForm77.Edit1Change(Sender: TObject);
begin
  inherited;
  if (Trim(Edit1.Text)<>'')and
  (Trim(Edit2.Text)<>'')and(Trim(ComboBox1.Text)<>'')and(Trim(ComboBox2.Text)<>'')and
      (Trim(Edit3.Text)<>'')and(Trim(Edit4.Text)<>'')and(Trim(Edit5.Text)<>'')and
      (Trim(ComboBox3.Text)<>'')and(Trim(ComboBox4.Text)<>'')and(aa = True)then
      BitBtn2.Enabled := True
```

```
    else
      BitBtn2.Enabled := False;
end;
```

将其他编辑框、组合框控件的 OnChange 事件调用上面的程序代码，如图 20.16 所示。

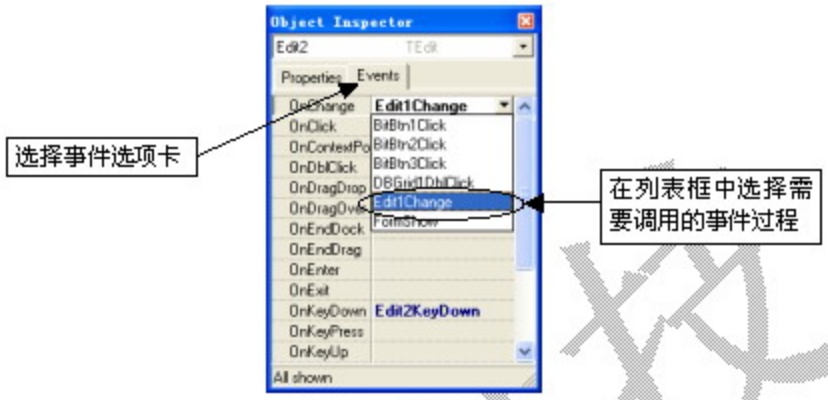


图 20.16　共享程序代码

20.8.5　图片的切换

在本章程序的人员调动、人员档案删除等模块中，作者没有使用按钮控件，而是使用图像控件实现数据的浏览、修改、删除等操作。在程序运行时，当用户按下图像控件时，会发现其颜色发生了变化。事实上，程序将当前图像控件隐藏了，然后将另一个图像控件显示出来，利用图像的切换实现其颜色的改变。同样的道理，当用户在图像控件上释放鼠标时，会将当前控件隐藏起来，将原来的图像控件显示出来。程序代码如下：

```
procedure TForm79.Image6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  inherited;
  Image6.Visible := False;
  Image5.Visible := True;
end;
procedure TForm79.Image6MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  inherited;
  Image5.Visible := False;
  Image6.Visible := True;
end;
```

20.8.6　控制按钮是否可用

在本章程序的部门信息管理模块中，在一个窗体中同时实现了部门信息的浏览、添加、修改、删除等功能。因此，窗体中也放置了相应的按钮。为了防止用户操作上的失误，需要为这些按钮设置逻辑关系。例如，用户在浏览数据时，“保存”按钮应该不可用，因为在浏览数据时，当前数据已在数据库中存在，不能再将其添加到数据库中。用户在添加数据

时，“修改”与“删除”按钮应不可用，因为当前数据在数据库中不存在，不能够修改和删除这些数据。

20.9 程序调试与错误处理

在开发应用程序时，总会遇到各种错误，例如，在本系统的实现过程中，就遇到了许多错误。在出现错误时，要仔细观察错误提示，根据错误提示查找错误形成原因。

在设计本系统的公司信息管理模块过程中，曾出现过如图 20.17 所示的错误提示。

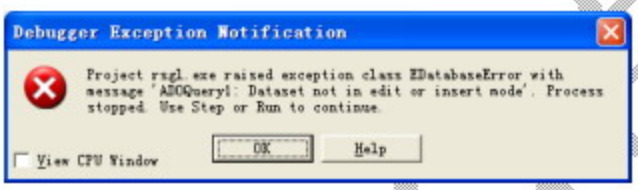


图 20.17 错误提示

从错误提示信息中可以看出是一个数据库错误，数据集没有处于编辑或插入状态。“修改”按钮的原始代码如下：

```
procedure TForm38.BitBtn1Click(Sender: TObject);
begin
  if Application.MessageBox('确实要修改公司信息吗？','提示',MB_YesNo)= ID_Yes then
  begin
    Data3.ADOConnection1.BeginTrans;
    Try
      Data3.ADOQuery1.FieldByName('公司名称').Value := Trim(Edit1.Text);
      Data3.ADOQuery1.FieldByName('公司简称').Value := Trim(Edit2.Text);
      Data3.ADOQuery1.FieldByName('地址').Value := Trim(Edit3.Text);
      Data3.ADOQuery1.FieldByName('电话').Value := Trim(MaskEdit1.Text);
      Data3.ADOQuery1.FieldByName('邮政编码').Value := Trim(Edit4.Text) ;
      Data3.ADOQuery1.FieldByName('网址').Value := Trim(Edit5.Text);
      Data3.ADOQuery1.FieldByName('电子邮件').Value := Trim(Edit6.Text);
      Data3.ADOQuery1.FieldByName('总经理').Value := Trim(Edit7.Text);
      Data3.ADOQuery1.Post;
      Data3.ADOConnection1.CommitTrans;
      BitBtn1.Enabled := False;
      Application.MessageBox('数据修改成功。','提示',64);
    Except
      Data3.ADOConnection1.RollbackTrans;
      Application.MessageBox('系统出错。','提示',64);
      Close;
    End;
  End
  else
    Self.OnShow(Sender);
end;
```

当使用断点进行跟踪时，发现程序在执行完“Data3.ADOQuery1.Post”语句后出现上面的错误提示，仔细检查代码，会发现“Data3.ADOQuery1.Post”语句之前不存在使数据集处于编辑或插入状态的语句，所以，上面的程序代码应改为：

```
procedure TForm38.BitBtn1Click(Sender: TObject);
begin
  if Application.MessageBox('确实要修改公司信息吗？','提示',MB_YesNo)= ID_Yes then
  begin
    Data3.ADOConnection1.BeginTrans;
    Try
      Data3.ADOQuery1.Edit; //使数据集处于编辑状态
      Data3.ADOQuery1.FieldByName('公司名称').Value := Trim(Edit1.Text);
      Data3.ADOQuery1.FieldByName('公司简称').Value := Trim(Edit2.Text);
      Data3.ADOQuery1.FieldByName('地址').Value := Trim(Edit3.Text);
      Data3.ADOQuery1.FieldByName('电话').Value := Trim(MaskEdit1.Text);
      Data3.ADOQuery1.FieldByName('邮政编码').Value := Trim(Edit4.Text) ;
      Data3.ADOQuery1.FieldByName('网址').Value := Trim(Edit5.Text);
      Data3.ADOQuery1.FieldByName('电子邮件').Value := Trim(Edit6.Text);
      Data3.ADOQuery1.FieldByName('总经理').Value := Trim(Edit7.Text);
      Data3.ADOQuery1.Post;
      Data3.ADOConnection1.CommitTrans;
      BitBtn1.Enabled := False;
      Application.MessageBox('数据修改成功。','提示',64);
    Except
      Data3.ADOConnection1.RollbackTrans;
      Application.MessageBox('系统出错。','提示',64);
      Close;
    End;
  end
  else
    Self.OnShow(Sender);
end;
```

20.10 程序设计清单

人事管理系统文件清单如表 20.10 所示。

表 20.10　　人事管理系统设计清单

文 件 名	文 件 类 型	说　明
Bmjcxxgl.dfm	窗体文件	部门信息管理窗体
Dataa.dfm	窗体文件	数据模块
Gsxxgl.dfm	窗体文件	公司信息管理窗体
MP.dfm	窗体文件	模板窗体
Rsglxt.dfm	窗体文件	主窗体

续表

文 件 名	文 件 类 型	说 明
Rylbxxgl.dfm	窗体文件	人员类别管理窗体
Ryxxgl.dfm	窗体文件	人员信息管理窗体
Scryda.dfm	窗体文件	删除人员档案窗体
Tjyusc.dfm	窗体文件	操作员管理窗体
Xgryda.dfm	窗体文件	修改人员档案窗体
Xtdl3.dfm	窗体文件	系统登录窗体
Ygddgl.dfm	窗体文件	员工调动管理窗体
Ygxxgl.dfm	窗体文件	员工信息管理窗体
Yhxxgl.dfm	窗体文件	银行信息管理窗体
Rsgl.dpr	工程文件	
Rsgl.exe	可执行文件	

20.11 技术、经验总结

20.11.1 技术总结

双击列表视图中的图标打开相应窗体。在本系统中，用户双击列表视图中的图标会打开相应的窗体，程序代码如下：

```
procedure TForm74.ListView1DblClick(Sender: TObject);
begin
  if ListView1.SelCount>0 then
  begin
    if ListView1.Selected.Caption = '公司信息管理' then
      N2.OnClick(Sender)
    else if ListView1.Selected.Caption = '部门信息管理' then
      N3.OnClick(Sender)
    else if ListView1.Selected.Caption = '人员类别管理' then
      N14.OnClick(Sender)
    else if ListView1.Selected.Caption = '银行信息管理' then
      N13.OnClick(Sender)
    else if ListView1.Selected.Caption = '员工信息管理' then
      N4.OnClick(Sender)
    else if ListView1.Selected.Caption = '员工调动管理' then
      N7.OnClick(Sender)
    else if ListView1.Selected.Caption = '操作员管理' then
      N8.OnClick(Sender);
  end;
end;
```

在上面代码中，注意条件语句“if ListView1.SelCount>0 then”的作用，如果去掉该语

句，运行程序，用户双击列表视图中的图标也会打开相应的窗体，但是，如果用户在列表视图中双击而没有在图标上双击，程序便会出现地址访问错误。

20.11.2 经验总结

在开发数据库应用程序时，如果需要修改或删除数据，在修改或删除数据之前，一定要提示用户是否真的要修改或删除数据，以防止用户不小心错误地修改或删除数据。

20.12 程序的编译与发行

本系统编译后形成的 EXE 文件为 1.35MB，要想使编译出来的 EXE 文件更小，在编译时可以采用运行期包，方法是在系统菜单“Project”下选择“Options”菜单项，会弹出一个工程选项对话框，打开“Packages”选项卡，如图 20.18 所示。

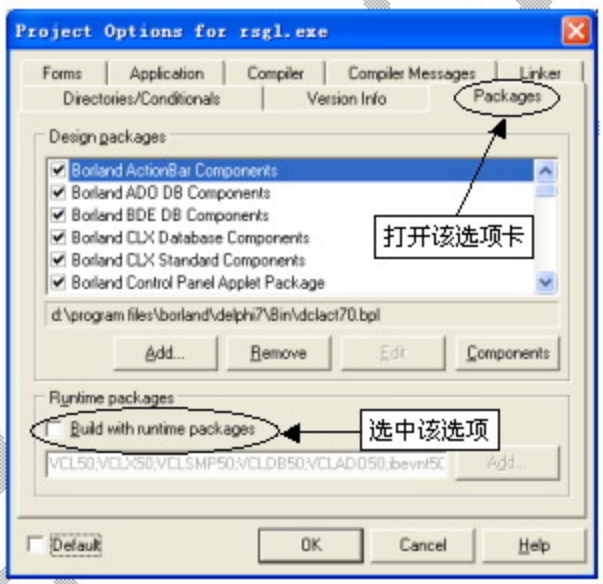


图 20.18 工程选项

在“Packages”选项卡中有一个“Build with runtime packages”选项，选中该选项，单击 OK 按钮完成设置，重新运行应用程序，会发现本系统的应用程序只有 362KB。但是，在发布应用程序时，需要额外发布所需 BPL 文件和 DLL 文件，程序才能正常运行。读者对此可能会产生疑问。在默认情况下，工程选项中的“Build with runtime packages”选项没有被选中，即系统链接时采用设计期包，生成的可执行文件包含了所有用到的 VCL 库文件，即 BPL 文件和 DLL 文件，所以编译出来的 EXE 文件比较大，而选中“Build with runtime packages”选项正是将程序所用到的库文件从 EXE 文件中分离出来。

如果要在其他机器上运行本系统，需要具备本系统的 EXE 文件，并且需要将人事管理系统数据库文件附加到 SQL Server 2000 中。

第21章 Delphi 在客户管理系统中的应用

21.1 客户管理系统概述

21.1.1 客户管理系统简介

当企业之间的产品差异与管理差异日渐缩小，企业内部资源的挖掘潜力不大，而企业面临的竞争更加激烈时，企业自然把目光投向了企业最重要的外部资源——客户资源的挖掘上来。客户管理是一种以客户为中心的经营理念。客户管理系统为企业实现客户管理信息化，它记录一个企业所有客户的相关信息，能够对客户进行信息跟踪，对客户反馈信息进行记录。企业管理人员通过客户管理系统能够了解客户的业务情况，能够根据客户反馈的信息及时地了解市场。

21.1.2 系统功能描述

- 实现客户基础信息录入，这一部分主要由操作员完成。
- 实现银行账户管理，为每一个客户建立银行账户。
- 设置系统登录，提高系统安全性。
- 实现客户通过电话完成订货功能。

21.2 系统分析

21.2.1 系统规划

系统规划阶段的主要工作是了解问题背景，决定系统开发的可行

性，进一步确定需求分析。完成系统规划一般需要经过 5 个步骤，即可行性研究、定义问题、搜集系统背景资料、了解现行工作、定义新的需求分析。可行性研究、定义问题、搜集系统背景资料、了解现行工作是企业管理人员和业务人员的工作，超出了本书的讨论范围，在此不做详细介绍，如果读者想了解更多这方面的信息，请参考专业的需求分析书籍。

根据上面系统功能的描述，定义需求分析如下。

1. 系统目标

将客户信息运用计算机信息系统管理，取代人工操作方式。

2. 系统功能

系统应具备以下处理功能。

- 客户信息管理

包含客户级别管理、客户区域管理、客户基础信息管理、客户信息反馈管理、客户信息查询管理、客户信息呼叫管理，其中在客户基础信息管理中，要求实现客户信息的浏览、添加、修改、删除。

- 商品信息管理

实现商品信息的添加、修改、删除。

- 操作员管理

实现操作员的添加、删除、权限设置。

- 银行账户管理
- 反馈信息查询管理

操作员通过录入或选择客户名称，程序能够显示客户的反馈信息，并且能够进行时间段查询。

3. 系统结构图

综合上面的系统功能描述，衡量实际的系统需求，现将系统结构划分为如图 21.1 所示的结构。

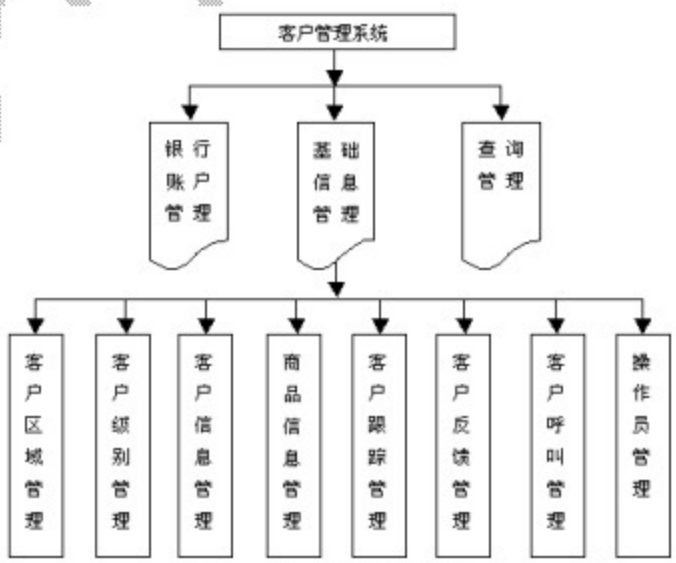


图 21.1　客户管理系统结构图

21.2.2 系统流程图

客户管理系统的流程图如图 21.2 所示。

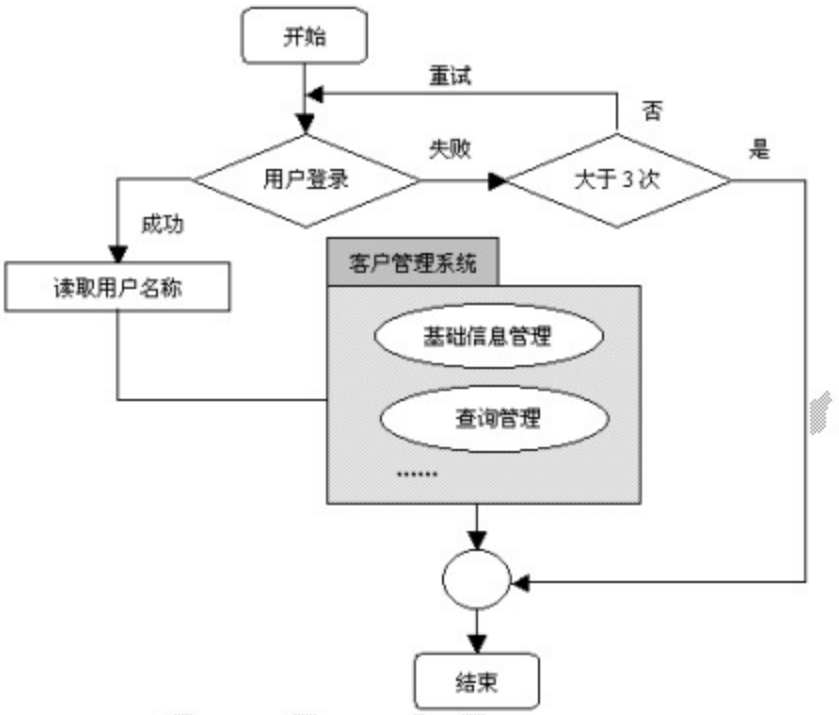


图 21.2 客户管理系统流程图

21.2.3 业务流程

客户管理系统的业务流程如图 21.3 所示。

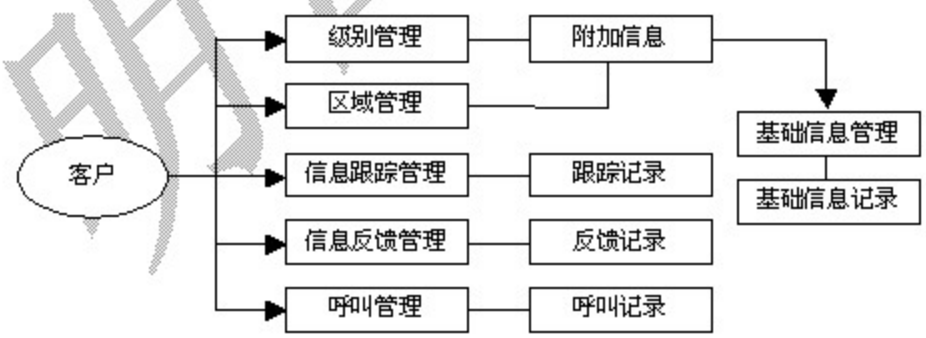


图 21.3 客户管理系统业务流程图

21.3 数据库设计

在确定程序的基本框架之后，便进入数据库设计阶段。数据库设计是系统设计的核心，合理的数据表结构不仅有利于程序的快速开发，而且有利于系统维护。

21.3.1 数据库概念设计

数据库概念设计是将数据库中数据的逻辑结构和特征以关系图的形式描述出来。客户管理系统的 E-R 图如图 21.4 所示。

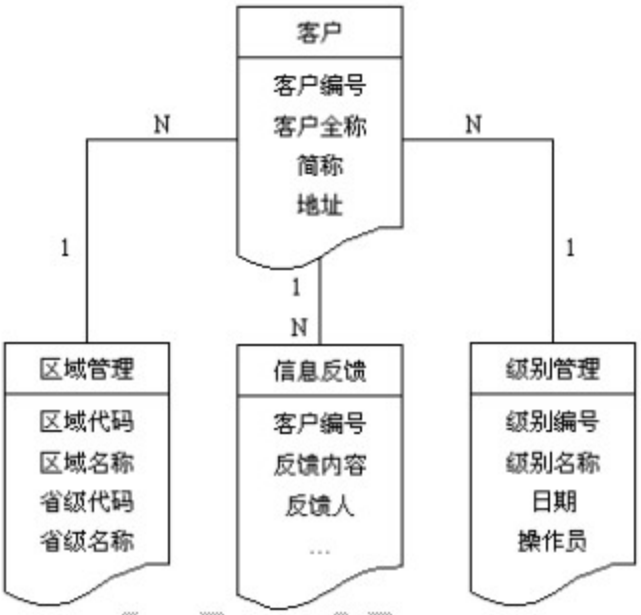


图 21.4 客户管理系统 E-R 图

21.3.2 数据库逻辑结构设计

客户管理系统共使用了 10 张数据表，分别为：客户基础信息表、银行账户管理表、客户集中呼叫票号表、商品基础信息表、客户集中呼叫表、客户信息跟踪表、客户信息反馈表、客户级别表、客户区域表、操作员信息表。由于篇幅关系，在此只列出使用较多的数据表结构（如表 21.1～21.4 所示），其他表结构请参考光盘。

表 21.1 商品基础信息表

字段名称	字段类型	主 键	外 键	是否为空	默 认 值
商品编号	Varchar(30)	是		否	
商品名称	Varchar(30)				
简称	Varchar(20)				
规格	Varchar(20)				
产地	Varchar(30)				
单位	Varchar(20)				
保质期	Int				
进价	Money				
批发价	Money				
零售价	Money				
备注	Varchar(60)				

表 21.2　　客户基础信息表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
客户编号	Varchar(30)	是		否	
客户全称	Varchar(30)				
简称	Varchar(20)				
地址	Varchar(30)				
电话	Varchar(30)				
传真	Varchar(30)				
联系人	Varchar(20)				
联系人电话	Varchar(30)				
开户银行	Varchar(30)				
纳税人	Varchar(20)				
邮箱地址	Varchar(30)				
邮政编码	Varchar(30)				
银行账号	Varchar(30)				
区域代码	Varchar(30)		是		
区域名称	Varchar(30)				
级别名称	Varchar(30)		是		

表 21.3　　操作员信息表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
操作员编号	Varchar(20)	是		否	
操作员姓名	Varchar(20)				
操作员密码	Varchar(20)				
操作员级别	Varchar(20)				
时间	DateTime				GetDate()

表 21.4　　客户信息跟踪表

字段名称	字段类型	主　键	外　键	是否为空	默 认 值
客户编号	Varchar(20)				
客户姓名	Varchar(30)				
客户动态	Text				
业务情况	Text				
时间	DateTime				GetDate()

21.4　处理过程分析

21.4.1　客户基础信息处理过程分析

客户基础信息管理主要是对客户基础信息的维护，即实现客户信息的添加、修改、删除、浏览等功能。客户基础信息管理业务流程如图 21.5 所示。

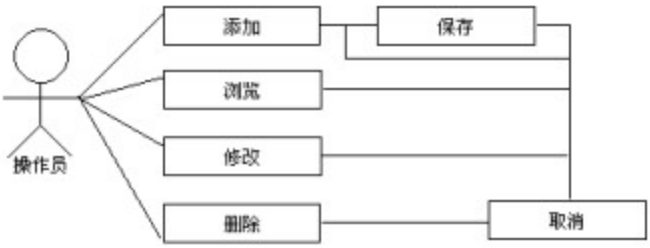


图 21.5　客户基础信息管理业务流程图

21.4.2　客户信息呼叫处理过程分析

客户信息呼叫是通过客户电话实现订货。要实现订货，就需要将客户及其所需商品信息保存到数据表中，所以，首先应该登记，添加单据编号，然后输入客户电话，通过电话调出客户相关信息，将信息添加到列表中，最后输入客户所需商品信息，将信息保存到数据表中。客户信息呼叫业务流程如图 21.6 所示。

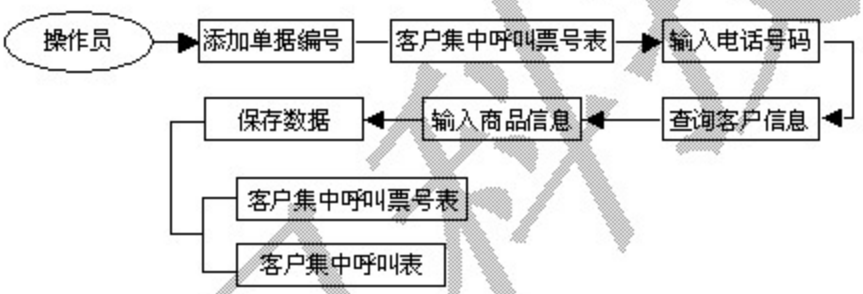


图 21.6　客户信息呼叫业务流程图

21.5　主窗体设计

主窗体设计步骤如下。

（1）创建一个窗体，在窗体中放置 TMainMenu、TTreeView、TImageList、TCoolBar、TToolBar、TStatusBar、TSpeedButton 等组件，如图 21.7 所示。



图 21.7　主窗体设计

（2）设置各主要组件属性如表 21.5 所示。

表 21.5　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form91	TForm	Caption	客户管理
		Font.Color	clBlue
		Font.Size	8
ListView1	TlistView	Images	ImageList1

（3）在窗体上双击 TImageList 组件，将弹出图像列表编辑框，单击 Add 按钮，可以将位图或图标加到 TImageList 中，如图 21.8 所示。



图 21.8　图像列表编辑框

（4）在窗体上双击 TStatusBar 组件，打开状态栏面板编辑器，单击 Add New 按钮添加面板。选中面板选项，通过修改对象观察器中的“Text”属性设置面板标题，效果如图 21.9 所示。

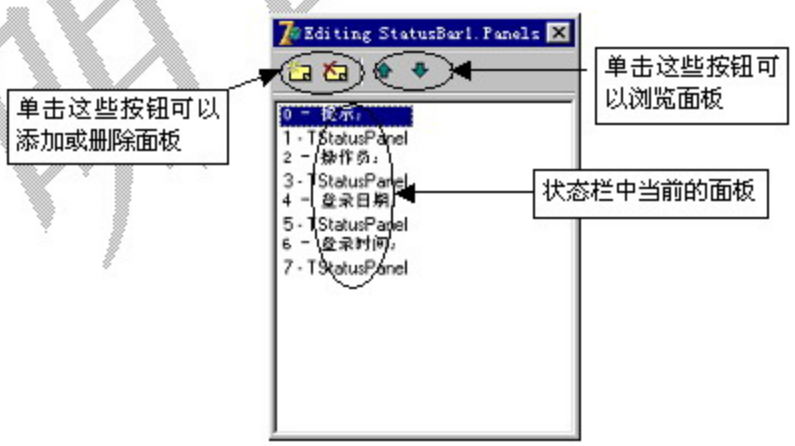


图 21.9　面板编辑器

（5）选中 TTreeView 组件，设置该组件的“Images”属性为“ImageList1”。双击该组件，打开项目编辑框，单击 New Item 按钮，创建项目，在“Text”选项中输入项目标题，在“Image Index”选项中输入图像索引。效果如图 21.10 所示。



图 21.10　项目编辑器

21.6　菜单设计

菜单设计步骤如下。

（1）在 Delphi 中提供了 TMainMenu 组件用来设计系统菜单，双击该组件，将打开菜单编辑窗口，如图 21.11 所示。

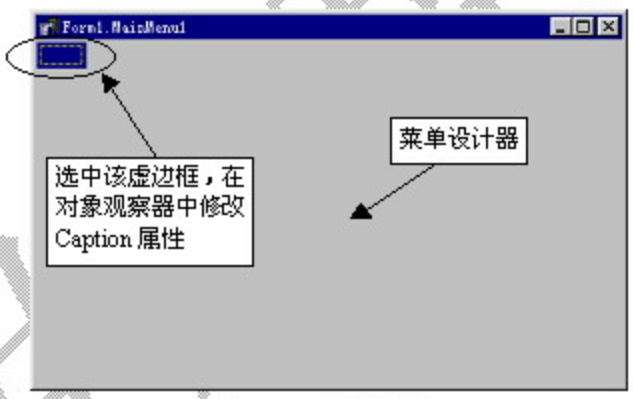


图 21.11　菜单编辑窗口

（2）弹出的菜单各有一个虚边框，选中该虚边框，在对象观察器中修改“Caption”属性，例如“基础信息管理(&W)”，单击菜单编辑窗口，效果如图 21.12 所示。

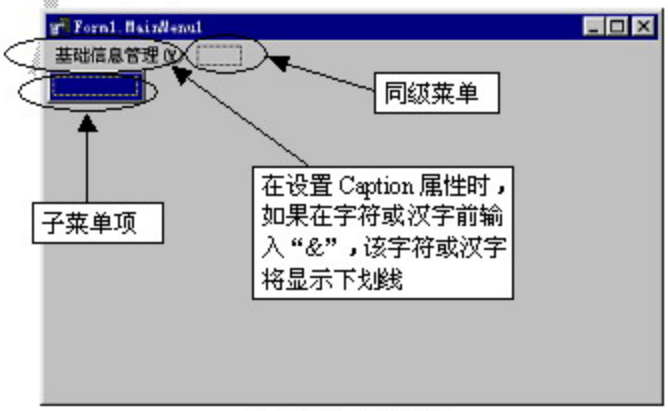


图 21.12　菜单设计

（3）本系统中使用的菜单属性设置如表 21.6 所示。

表 21.6　　菜单属性设置

对 象 名	对 象 类 型	属　　性	值
MainMenu1	TMainMenu	Images	ImageList1
W1	TmenuItem	Caption	基础信息管理(&W)
N9	TmenuItem	Caption	客户区域管理
		ImageIndex	2
N13	TmenuItem	Caption	客户级别管理
		ImageIndex	3
N5	TmenuItem	Caption	客户基础信息管理
		ImageIndex	7
N1	TmenuItem	Caption	商品基础信息管理
		ImageIndex	1
N14	TmenuItem	Caption	客户信息跟踪
		ImageIndex	4
N15	TmenuItem	Caption	客户信息反馈
		ImageIndex	5
N16	TmenuItem	Caption	客户信息呼叫
		ImageIndex	6
N2	TmenuItem	Caption	操作员管理
		ImageIndex	5
N17	TmenuItem	Caption	银行账户管理(&Y)
Q1	TmenuItem	Caption	查询管理(&Q)

（4）菜单设计效果如图 21.13 所示。



图 21.13　菜单界面

21.7　工具栏设计

在 Delphi 中提供了 TTooBar 类用来设计工具栏，但是，在 Delphi 早期的版本中，例如，Delphi 2 中并没有提供 TTooBar 类，当时，Delphi 程序员要设计一个工具栏需要在窗体上放置一个控制面板，将它的“Align”属性设置为“alTop”，然后向控制面板中添加加速按

钮（TSpeedButton）并将它们排列好。对于一般的 Windows 应用程序，工具栏上的按钮只是重复菜单中的功能，所以，在工具栏按钮的 OnClick 事件中可以调用与它功能相同的菜单项的 OnClick 事件。例如：

```
procedure TForm1.ToolButton1Click(Sender: TObject);
begin
  N9Click(ToolButton1);
end;
```

也可以选中工具栏按钮，在对象观察器的事件选项卡中设置“OnClick”事件为“N9Click”，如图 21.14 所示。

下面以创建本系统工具栏为例介绍如何设计工具栏。

（1）在窗体中选中 TToolBar 组件，设置 TToolBar 的“EdgeBorders.ebTop”属性为“False”。在 TToolBar 组件中放置 TSpeedButton 组件，设置 TSpeedButton 组件的“Glyph”属性与“Caption”属性。

（2）用鼠标右键单击工具栏组件，会弹出快捷菜单，如图 21.15 所示。



图 21.14　对象观察器



图 21.15　快捷菜单

（3）单击 New Button 按钮，向工具栏中添加按钮，选中该按钮，在对象观察器中修改“Style”属性为“tbsDivider”，然后调整按钮的宽度，为工具栏设置分隔条。效果如图 21.16 所示。



图 21.16　客户管理系统工具栏

21.8　数据模块设计

数据模块设计步骤如下。

（1）启动 Dephi 7，打开工程“wlgl”，单击“File”/“New”/“Data Module”菜单项，创建数据模块。

（2）选中数据模块，在对象观察器中修改“Name”属性为“Datam”。

（3）在数据模块上放置 TADOConnection 组件、TADOQuery 组件，如图 21.17 所示。



图 21.17　数据模块

（4）设置各主要组件属性如表 21.7 所示。

表 21.7　　组件属性设置

对　象　名	对 象 类 型	属　　性	值
ADOConnection1	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=客户管理系统
ADOQuery1	TADOQuery	Connection	ADOConnection1
ADOQuery2	TADOQuery	Connection	ADOConnection1
ADOQuery4	TADOQuery	Connection	ADOConnection1

21.9　程序设计与编码

21.9.1　客户区域管理程序设计

1. 概述

客户区域管理主要实现客户区域信息的添加、修改、浏览、删除等操作。用户在添加客户基础信息时，如果客户区域信息不存在，需要添加客户区域信息。在本系统的客户区域管理中，在窗体显示时，会将客户区域表中的数据显示在窗体中，用户单击“上一条”、“下一条”按钮可以浏览数据，单击“添加”按钮，程序会自动生成区域代码，要求用户输入区域名称。输入区域名称后，按 Enter 键，下拉列表框会获得焦点，要求用户选择省级名称。选择省级名称后，按 Enter 键会显示省级代码，单击“保存”按钮会将数据保存到数据表中。客户区域管理运行结果如图 21.18 所示。



图 21.18　客户区域管理运行结果

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox 等组件。

（2）设置各主要组件属性如表 21.8 所示。

表 21.8　　组件属性

对 象 名	对 象 类 型	属　性	值
Form23	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
BitBtn4	TBitBtn	Enabled	False

3. 程序相关代码

当窗体显示时，进行界面初始化，并从客户区域数据表中读取信息，将信息显示在界面上。

```
procedure TForm23.FormShow(Sender: TObject);
begin
  with datam.ADOQuery2 do
  begin
    Close; //关闭数据集
    SQL.Clear;//清空SQL语句
    SQL.Add('select * from 客户区域表');//添加SQL语句
    Open; //打开数据集
  end;
  //如果数据表中没有数据，进行提示，并将一些按钮不可用
  If datam.ADOQuery2.FieldByName('区域代码').Value = null then
  begin
    Application.MessageBox('该表数据为空，请向该表中插入数据。','提示',64);
    BitBtn6.Enabled := False; //使BitBtn6不可用
    BitBtn7.Enabled := False;
    BitBtn1.Enabled := False;
    BitBtn2.Enabled := False;
  end
  else
  begin
    //调用Clear过程将界面上的数据初始化
    Clear;
    //调用EditValue过程将数据集当前记录显示在界面中
    EditValue;
  end;
end;
```

用户在 Edit2 上按 Enter 键时，如果 Edit2 不为空，ComboBox1 将获得焦点。

```
procedure TForm23.Edit2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
```

```
begin
  if Key = VK_ReTurn then
    if Trim(Edit2.Text)<>'' then
    begin
      //使ComboBox1可用
      ComboBox1.Enabled := True;
      ComboBox1.SetFocus;
    end;
end;
```

当用户在 ComboBox1 上按 Enter 键时，会自动生成省级代码。程序首先判断省级名称是否为空，如果为空则取消操作，如果不为空，以省级名称为条件从客户区域表中查询省级代码，如果有数据返回，则读取省级代码，否则生成一个新的省级代码。

在生成省级代码时，首先指定字符串前缀，由指定字符串（例如“P”）与字符串格式的系统日期（yyyymm 格式，例如，“200504”）组成，然后确定代码序号。在确定代码序号时，首先读取当前数据表中最大的省级代码（例如，200502007），截取其序号（007），使其转换为整数，加 1（8），再转换成字符串格式（008），即新的序号。将字符串前缀与序号组合，形成新的省级代码（P200504008）。

```
procedure TForm23.ComboBox1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
  s,m: String;
  i: integer;
begin
  if Key = VK_ReTurn then
    if Trim(ComboBox1.Text)<> '' then
    begin
      with datam.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 客户区域表 where 省级名称 = :a');
        Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
        Open;
      end;
      if datam.ADOQuery1.RecordCount>0 then
      begin
        Label6.Caption :=  datam.ADOQuery1.FieldByName('省级代码').Value;
        Edit1.Enabled := True;
        Edit1.SetFocus;
        BitBtn4.Enabled := True;
      end
      else
      begin
        s:= 'P'+FormatDateTime('yyyymm',Now());
        With datam.ADOQuery1 do
```

```
            begin
              Close;
              SQL.Clear;
              SQL.Add('select max(省级代码) as ss From 客户区域表 ');
              Open;
            end;
            If datam.ADOQuery1.FieldByName('ss').Value = null then
              s := s + '001'
            else
            begin
              m:= Trim(datam.ADOQuery1.FieldByName('ss').Value) ;
              i:= StrToInt(Trim(Copy(m,8,10))) ;
              if i<9 then
                s:= s + '00'+IntToStr(i +1)
              else if i<99 then
                s:= s + '0'+IntToStr(i +1)
              else
                s:= s +IntToStr(i +1);
            end;
            Label6.Caption := s;
            if (Trim(Label9.Caption)='')and(Trim(Label12.Caption)='')then
              Label14.Caption := '省级代理';
            Edit1.Enabled := True;
            Edit1.SetFocus;
            BitBtn4.Enabled := True;
          end;
        end;
    end;
```

处理 ComboBox1 的 OnChange 事件，当 ComboBox1 中当前选项变化时，执行界面清理工作。

```
procedure TForm23.ComboBox1Change(Sender: TObject);
begin
  Edit1.Clear;
  Edit3.Clear;
  Label14.Caption := '';
  Label6.Caption := '';
  Label9.Caption := '';
  Label12.Caption := '';
  BitBtn4.Enabled := False;
end;
```

下面代码主要实现在用户修改市级名称时，清空县级名称、县级代码、市级代码。

```
procedure TForm23.Edit1Change(Sender: TObject);
begin
  Edit3.Clear;
  Label14.Caption := '省级代理';
  Label9.Caption := '';
```

```
  Label12.Caption := '';
end;
```

下面代码实现当县级名称改变时，清空县级代码。

```
procedure TForm23.Edit3Change(Sender: TObject);
begin
  Label14.Caption := '市级代理';
  Label12.Caption := '';
end;
```

当区域名称改变时，清空区域代码、市级名称、县级名称、市级代码、县级代码。

```
procedure TForm23.Edit2Change(Sender: TObject);
begin
  Edit1.Enabled := False;
  Edit3.Enabled := False;
  Edit1.Clear;
  Edit3.Clear;
  Label14.Caption := '';
  Label6.Caption := '';
  Label9.Caption := '';
  Label12.Caption := '';
end;
```

用户单击“保存”按钮，会将用户录入的信息保存到客户区域表中。

```
procedure TForm23.BitBtn4Click(Sender: TObject);
begin
  if Trim(Label14.Caption)<>'' then
  begin
    Try //异常保护
      With datam.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('Insert 客户区域表 values (:a ,:b ,:c,:d,:e,:f,:g,:h,:i)');
        Parameters.ParamByName('a').Value := Trim(Label4.Caption);
        Parameters.ParamByName('b').Value := Trim(Edit2.Text);
        Parameters.ParamByName('c').Value := Trim(Label6.Caption);
        Parameters.ParamByName('d').Value := Trim(Combobox1.Text);
        Parameters.ParamByName('e').Value := Trim(Label9.Caption);
        Parameters.ParamByName('f').Value := Trim(Edit1.Text);
        Parameters.ParamByName('g').Value := Trim(Label12.Caption);
        Parameters.ParamByName('h').Value := Trim(Edit3.Text);
        if Trim(Label14.Caption)='省级代理' then
          Parameters.ParamByName('i').Value := 1
        else if Trim(Label14.Caption)='市级代理' then
          Parameters.ParamByName('i').Value := 2
        else
          Parameters.ParamByName('i').Value := 3;
        ExecSQL;
```

```
            end;
            Application.MessageBox('添加成功。','提示',64);
            BitBtn7.OnClick(Sender);
          Except //处理异常
            Application.MessageBox('系统出错。','提示',64);
            Close;
          end;
        end;
      end;
```

21.9.2 客户级别管理程序设计

1. 概述

客户级别管理主要实现客户级别信息的添加和删除。客户级别信息是客户的附加信息，它在一定程度上反映了客户企业的规模。用户在添加客户基础信息时，如果客户级别信息不存在，需要添加客户级别信息。在本系统的客户级别管理中，用户单击“添加”按钮，程序会显示系统日期和操作员，同时生成级别编号。输入级别名称后，按 Enter 键，备注编辑框会获得焦点，输入备注后，单击“保存”按钮会将数据保存到数据表中。客户区域管理运行结果如图 21.19 所示。

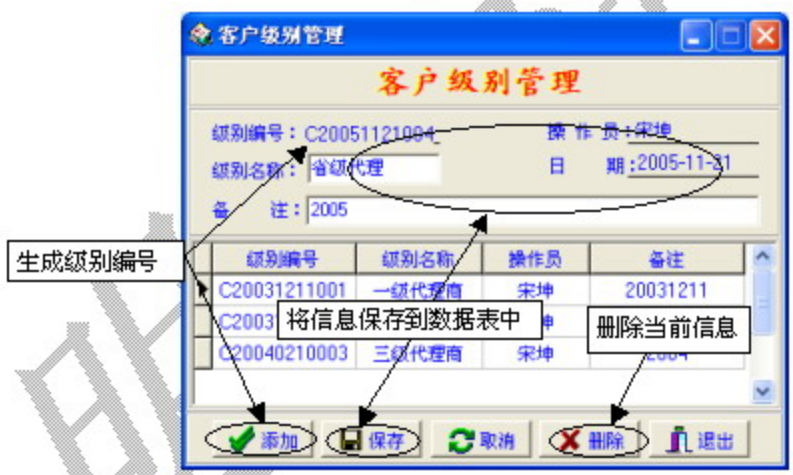


图 21.19　客户级别管理窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TDBGrid、TDataSource、TShape 等组件。

（2）设置各主要组件属性如表 21.9 所示。

表 21.9　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form25	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog

续表

对 象 名	对 象 类 型	属 性	值
BitBtn3	TBitBtn	Enabled	False
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

在窗体显示时会读取系统时间和操作员，同时将客户级别表信息显示在列表中。

```
procedure TForm25.FormShow(Sender: TObject);
begin
  // 显示系统时间和操作员
  Label5.Caption := FormatDateTime('yyyy-mm-dd',Now);
  Label6.Caption := czymc; //czymc是主窗体单元定义的一个全局变量
  with datam.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 客户级别表');
    Open;
  end;
  if datam.ADOQuery1.RecordCount>0 then
    DataSource1.DataSet := datam.ADOQuery1;
end;
```

单击“添加”按钮，程序自动生成级别编号。级别编号的生成方法与上一节介绍的省级编号的生成方法一致，在此不做过多介绍。

```
procedure TForm25.BitBtn1Click(Sender: TObject);
var
  s,m: String;
  i: integer;
begin
  s:= 'C'+FormatDateTime('yyyymmdd',Now());
  With datam.ADOQuery2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select max(级别编号) as ss From 客户级别表 ');
    Open;
  end;
  If datam.ADOQuery2.FieldByName('ss').Value = null then
    s := s + '001'
  else
  begin
    m:= Trim(datam.ADOQuery2.FieldByName('ss').Value) ;
    i:= StrToInt(Trim(Copy(m,10,5))) ;
    if i<9 then
      s:= s + '00'+ InttoStr(i +1)
```

```
    else if i<99 then
      s:= s + '0'+ InttoStr(i +1)
    else
      s:= s +InttoStr(i+1);
  end;
  Label7.Caption := s;
  Label6.Caption := czymc;
  Label5.Caption := FormatDateTime('yyyy-mm-dd',Now);
  //使Edit1获得焦点
  Edit1.SetFocus;
  BitBtn3.Enabled := True;
end;
```

单击“保存”按钮，会将用户信息保存到数据表中。程序首先判断基本信息是否为空，如果为空，则退出操作；如果不为空，向客户级别表中插入数据，同时刷新窗口中的数据。

```
procedure TForm25.BitBtn3Click(Sender: TObject);
begin
  //只有各项不为空，才能向数据表中插入数据
  if (Trim(Edit1.Text)<>'')and(Trim(Edit2.Text)<>'')and(Trim(Label5.Caption)<>'')and
    (Trim(Label6.Caption)<>'')and(Trim(Label7.Caption)<>'')then
  begin
    Try
      with datam.ADOQuery2 do
      begin
        Close;
        SQL.Clear;
        //Default为默认值，是创建客户级别表时设置的默认值
        SQL.Add('Insert  客户级别表 values (:a ,:b,Default ,:d,:e)');
        Parameters.ParamByName('a').Value := Trim(Label7.Caption);
        Parameters.ParamByName('b').Value := Trim(Edit1.Text);
        Parameters.ParamByName('d').Value := Trim(Label6.Caption);
        Parameters.ParamByName('e').Value := Trim(Edit2.Text);
        ExecSQL;//执行SQL语句
      end;
      Application.MessageBox('添加成功。','提示',64);
      BitBtn5.OnClick(Sender); //添加成功后清空数据
      with datam.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from  客户级别表');
        Open;
      end;
      if datam.ADOQuery1.RecordCount>0 then
        DataSource1.DataSet := datam.ADOQuery1;
          BitBtn3.Enabled := False;
      BitBtn1.SetFocus;
```

```
        Except
          //产生异常，将进行提示并关闭窗体
          Application.MessageBox('系统出错。','提示',64);
          Close;
        end;
      end;
    end;
```

单击“删除”按钮将从数据表中删除当前信息，在删除信息时，程序会给出提示，让用户确认，防止误删数据。

```
procedure TForm25.BitBtn4Click(Sender: TObject);
begin
  if datam.ADOQuery1.RecordCount>0 then
    if Application.MessageBox(Pchar('确实要删除级别编号为'+
      Trim(datam.ADOQuery1.FieldByName('级别编号').Value) +'的级别信息吗？'),提示
        ',MB_YesNo)= ID_Yes then
      datam.ADOQuery1.Delete;
end;
```

21.9.3 客户基础信息管理程序设计

1. 概述

客户基础信息管理主要实现客户基础信息的添加、修改、浏览、删除等操作。它是客户管理系统的核心。客户区域管理和客户级别管理都是为实现客户基础信息管理而设计的。在窗体显示时，将客户信息显示在界面上，让用户浏览、修改、删除客户信息，如果数据表中没有客户信息，“修改”、“删除”等按钮将不可用，同时提示需要向客户基础信息表中输入数据。当用户单击“添加”按钮时，程序会生成客户编号，在用户输入完数据时，“保存”按钮将会变为可用，允许用户保存数据。单击“删除”按钮会弹出提示对话框，让用户确认是否删除当前行数据。客户基础信息管理运行结果如图 21.20 所示。

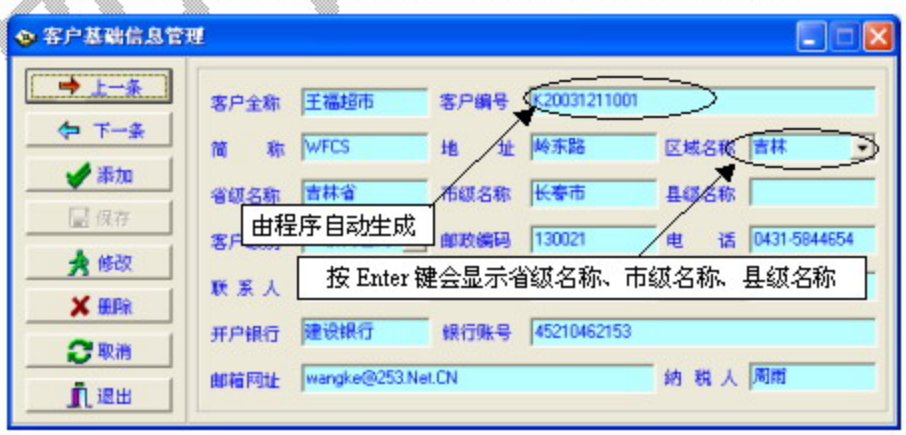


图 21.20　客户基础信息管理窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TDBGrid、

TDataSource、TShape、TMaskEdit、TDataSource、TComboBox、TDBGrid 等组件。

（2）设置各主要组件属性如表 21.10 所示。

表 21.10　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form9	TForm	Font.Color	clBlue
		Font.Size	10
		BorderStyle	bsDialog
BitBtn4	TBitBtn	Enabled	False
Edit1	TEdit	ReadOnly	True
DBGrid1	TDBGrid	DataSource	DataSource1
		Visible	False

3. 程序相关代码

下面是“保存”按钮的单击事件代码。在进行保存操作时，程序首先检查客户信息是否重复（不用担心客户信息为空，因为客户信息为空时，按钮不可用），如果重复进行提示，并退出，然后检查客户级别是否合法（是否在客户级别表中存在），如果不合法，进行提示，并退出。在经过了商品的检查之后，将数据保存到客户基础信息表中。

```
procedure TForm9.BitBtn4Click(Sender: TObject);
begin
  With datam.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 客户基础信息表 where 客户编号 = :a');
    Parameters.ParamByName('a').Value := Trim(Edit1.Text);
    Open;
  end;
  if datam.ADOQuery1.RecordCount > 0 then
  begin
    Application.MessageBox('该客户已经存在。','提示',0+64);
    Exit;
  end;
  With datam.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 客户级别表 where 级别名称 = :a');
    Parameters.ParamByName('a').Value := Trim(ComboBox2.Text);
    Open;
  end;
  if datam.ADOQuery1.RecordCount <1 then
  begin
    Application.MessageBox('客户级别不存在。','提示',64);
    ComboBox2.SetFocus;
```

```
        Exit;
      end;
      Try
      With datam.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('Insert 客户基础信息表 values
        ( :a,:b,:c,:d,:e,:f,:g,:h,:i,:j,:k,:l,:m,:n,:o,:p )');
        Parameters.ParamByName('a').Value := Trim(Edit1.Text);
        Parameters.ParamByName('b').Value := Trim(Edit2.Text);
        Parameters.ParamByName('c').Value := Trim(Edit3.Text);
        Parameters.ParamByName('d').Value := Trim(Edit4.Text);
        Parameters.ParamByName('e').Value := Trim(MaskEdit1.Text);
        Parameters.ParamByName('f').Value := Trim(Edit11.Text);
        Parameters.ParamByName('g').Value := Trim(Edit12.Text);
        Parameters.ParamByName('h').Value := Trim(Edit7.Text);
        Parameters.ParamByName('i').Value := Trim(Edit8.Text);
        Parameters.ParamByName('j').Value := Trim(Edit9.Text);
        Parameters.ParamByName('k').Value := Trim(Edit13.Text);
        Parameters.ParamByName('l').Value := Trim(Edit10.Text);
        Parameters.ParamByName('m').Value := Trim(Edit14.Text);
        Parameters.ParamByName('n').Value := Trim(qydm);
        Parameters.ParamByName('o').Value := Trim(ComboBox1.Text);
        Parameters.ParamByName('p').Value := Trim(ComboBox2.Text);
        ExecSQL;
      end;
      Application.MessageBox('保存成功。','提示',64);
      BitBtn1.Enabled := True;
      BitBtn2.Enabled := True;
      BitBtn4.Enabled := False;
      BitBtn7.OnClick(Sender);
      ShowDate;
      aa := False;
      Except
      Application.MessageBox('系统出错。','提示',0+64);
      Close;
      end;
    end;
```

下面代码确定“保存”按钮是否可用，只有编辑框中的内容不为空，“保存”按钮才可用，否则不可用，其他的编辑控件和列表框控件的 ONChange 事件都调用 Edit1 的 OnChange 事件处理函数。

```
procedure TForm9.Edit1Change(Sender: TObject);
begin
  if (Trim(Edit1.Text)<>'')and(Trim(Edit2.Text)<>'')and(Trim(Edit3.Text)<>'')and
    (Trim(Edit4.Text)<>'')and(Trim(ComboBox1.Text)<>'')and(Trim(Edit5.Text)<>'')and
```

```
    (Trim(ComboBox2.Text)<>'')and(Trim(Edit10.Text)<>'')and(Trim(MaskEdit1.Text)<>'')and
    (Trim(Edit12.Text)<>'')and(Trim(Edit7.Text)<>'')and(Trim(Edit11.Text)<>'')
    and(Trim(Edit14.Text)<>'')and(Trim(Edit8.Text)<>'')and(Trim(Edit9.Text)<>'')and(Trim
    (Edit13.Text)<>'')and(aa = True)then
    BitBtn4.Enabled := True
  else
    BitBtn4.Enabled := False;
end;
```

防止输入非法字符，在 Edit10 中只能输入整数。

```
procedure TForm9.Edit10KeyPress(Sender: TObject; var Key: Char);
var
  m: Boolean;
begin
  // #8为<Backspace>键，#48～#57为数字0～9
  m := (key <#8)or(Key>#8)and(key <#48)or(key>#57);
  if m then
    key := #0;
end;
```

下面代码实现在窗体显示时向 ComboBox1 中添加区域名称，向 ComboBox2 中添加级别名称，并将客户基础信息表中首行数据显示在界面中。

```
procedure TForm9.FormShow(Sender: TObject);
begin
  with datam.ADOQuery2 do //查询客户级别
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from  客户级别表');
    Open;
  end;
  if datam.ADOQuery2.RecordCount>0 then
  begin
    while Not datam.ADOQuery2.Eof   do //利用循环向组合框中添加客户级别
    begin
      ComboBox2.Items.Add(datam.ADOQuery2.FieldByName('级别名称').Value);
      datam.ADOQuery2.Next;
    end;
  end;
  with datam.ADOQuery2 do //查询客户区域
  begin
    Close;
    SQL.Clear;
    SQL.Add('select distinct  区域名称  from  客户区域表');
    Open;
  end;
  if datam.ADOQuery2.RecordCount>0 then
```

```
    while Not datam.ADOQuery2.Eof do //利用循环向组合框中添加区域信息
    begin
      ComboBox1.Items.Add(datam.ADOQuery2.FieldByName('区域名称').Value);
      datam.ADOQuery2.Next;
    end;
    with datam.ADOQuery2 do //查询客户信息
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from  客户基础信息表');
      Open;
    end;
    If datam.ADOQuery2.FieldByName('客户编号').Value = null then
    begin
      Application.MessageBox('该表数据为空，请向该表中插入数据。','提示',64);
      BitBtn5.Enabled := False;
      BitBtn6.Enabled := False;
      BitBtn7.Enabled := False;
      BitBtn1.Enabled := False;
      BitBtn2.Enabled := False;
    end
    else
    EditValue;//将当前记录集信息显示在界面中
  end;
```

下面代码主要实现浏览数据，单击“上一条”按钮会将上一条记录显示出来，单击“下一条”按钮会将下一条记录显示出来。

```
  procedure TForm9.BitBtn1Click(Sender: TObject);
  begin
    DisableComboBox;
    if Not datam.ADOQuery2.Bof   then
    begin
      datam.ADOQuery2.Prior;
      EditValue;
    end
    else
      BitBtn1.Enabled := False;
      BitBtn2.Enabled := True;
      BitBtn5.Enabled := True;
      BitBtn6.Enabled := True;
  end;
  procedure TForm9.BitBtn2Click(Sender: TObject);
  begin
    DisableComboBox;
    if Not datam.ADOQuery2.Eof then
    begin
      datam.ADOQuery2.Next;
```

```
    EditValue;
  end
  else
    BitBtn2.Enabled := False;
    BitBtn1.Enabled := True;
  BitBtn5.Enabled := True;
  BitBtn6.Enabled := True;
end;
```

以下是“修改”按钮的单击事件代码。程序首先判断当前记录集信息是否为空，如果为空，“修改”、“上一条”、“下一条”按钮不可用，并退出修改操作。如果当前记录集信息不为空，提示是否修改数据，选择“是”，程序将继续检查“省级名称”、“客户级别”是否为空，“客户级别”是否存在，如果“省级名称”、“客户级别”为空或者“客户级别”不存在，将退出修改操作，否则开始修改数据。

```
procedure TForm9.BitBtn5Click(Sender: TObject);
begin
  DisableComboBox;
  if datam.ADOQuery2.FieldByName('客户编号').Value = null then
  begin
    BitBtn5.Enabled := False;
    BitBtn1.Enabled := False;
    BitBtn2.Enabled := False;
    Exit;
  end;
  Try
  if Application.MessageBox('确实要修改该条记录吗？','提示',MB_YESNO )= ID_Yes then
  begin
    if Trim(Edit5.Text)='' then
    begin
      Application.MessageBox('省级名称不能为空。','提示',64);
      Exit;
    end;
    if Trim(ComboBox2.Text)='' then
    begin
      Application.MessageBox('客户级别不能为空。','提示',64);
      ComboBox2.SetFocus;
      Exit;
    end;
    with datam.ADOQuery4 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 客户级别表 where 级别名称 = :a');
      Parameters.ParamByName('a').Value := Trim(ComboBox2.Text);
      Open;
    end;
    if datam.ADOQuery4.RecordCount<1 then
```

```
        begin
          Application.MessageBox('该级别不存在。','提示',64);
          ComboBox2.Text := '';
          ComboBox2.SetFocus;
          Exit;
        end;
        with datam.ADOQuery2 do
        begin
          Edit;
          FieldByName('客户全称').Value := Edit2.Text ;
          FieldByName('简称').Value := Edit3.Text ;
          FieldByName('地址').Value := Edit4.Text ;
          FieldByName('级别名称').Value := Trim(ComboBox2.Text);
          FieldByName('区域代码').Value := Trim(qydm);
          FieldByName('区域名称').Value := Trim(ComboBox1.Text);
          FieldByName('邮政编码').Value := Edit10.Text ;
          FieldByName('电话').Value := MaskEdit1.Text ;
          FieldByName('传真').Value := Edit11.Text ;
          FieldByName('联系人').Value := Edit12.Text ;
          FieldByName('联系人电话').Value := Edit7.Text ;
          FieldByName('开户银行').Value := Edit8.Text ;
          FieldByName('纳税人').Value := Edit9.Text ;
          FieldByName('邮箱地址').Value := Edit13.Text ;
          FieldByName('银行账号').Value := Edit14.Text ;
          Post;
        end;
        Application.MessageBox('修改成功','提示',0+64);
        ShowDate;
      end
      else
        EditValue;
      Except
      Application.MessageBox('系统出错','提示',0+64);
      Close;
      end;
    end;
```

21.9.4 客户信息呼叫管理程序设计

1. 概述

客户信息呼叫能够通过客户电话完成订货。当用户输入客户的电话号码时，程序能够调出客户的相关信息，并将这些信息显示在列表中，用户在列表中输入客户需要的商品，按 Enter 键会将商品编号等信息显示出来，用户在商品名称字段按 PageDown 键，会将所有商品信息显示在一个列表中，可以在列表中选择商品。用户想要在列表中添加新行，可以在列表中按“↓”键，前提是当前行数据不能为空。当用户在列表中按 Delete 键时，会删除当前行数据。单击“保存”按钮，会将客户及其需要的商品信息保存到数据库中。客户

信息管理运行结果如图 21.21 所示。

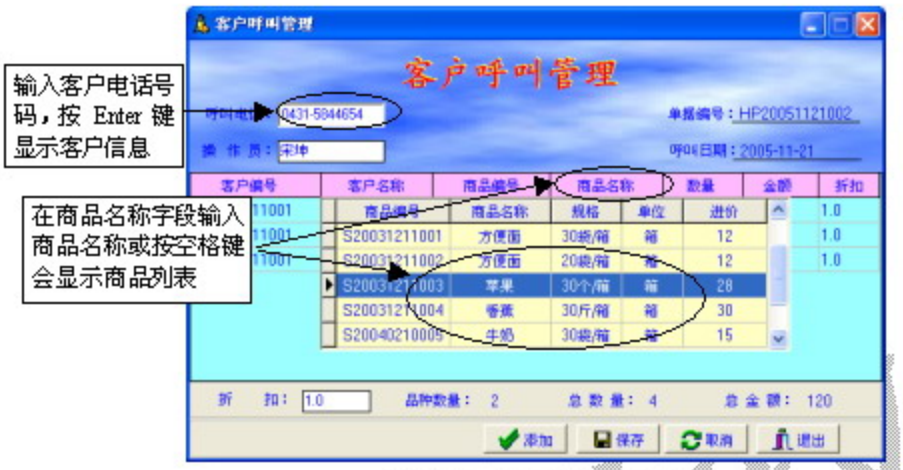


图 21.21　客户呼叫管理窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TImage、TStringGrid、TMaskEdit、TShape 、TDataSource、TDBGrid 等组件。

（2）设置各主要组件属性如表 21.11 所示。

表 21.11　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form30	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
BitBtn8	TBitBtn	Kind	bkClose
DBGrid1	TDBGrid	DataSource	DataSource1
		Visible	False

3. 程序相关代码

用户输入客户电话后按 Enter 键会将客户信息显示在列表中。程序首先确定“客户电话”、“单据编号”、“呼叫时间”是否为空，如果为空，退出操作，否则以“客户电话”为条件从客户基础信息表中查询数据，如果有数据返回，将其显示在表格中，否则提示客户信息不存在。

```
procedure TForm30.MaskEdit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = Vk_ReTurn then
    if Trim(MaskEdit1.Text)<>'' then //客户电话是否为空
    begin
      if (Trim(Label7.Caption)='')or(Trim(Label10.Caption)='')then
      //单据编号或呼叫时间是否为空
        begin
```

```
            Application.MessageBox('请添加单据编号或呼叫时间。','提示',64);
            BitBtn3.SetFocus;
            Exit;
          end;
          with datam.ADOQuery1 do
          begin
            Close;
            SQL.Clear;
            SQL.Add('select * from 客户基础信息表 where 电话 = :a');
            Parameters.ParamByName('a').Value := Trim(MaskEdit1.Text);
            Open;
          end;
          if datam.ADOQuery1.RecordCount>0 then
          begin
            MaskEdit1.ReadOnly := True;
            StringGrid1.Cells[6,StringGrid1.RowCount-1]:= '1.0';
            StringGrid1.Cells[0,StringGrid1.RowCount-1] := datam.ADOQuery1.FieldByName('客户
                编号').Value;
            StringGrid1.Cells[1,StringGrid1.RowCount-1] := datam.ADOQuery1.FieldByName('客户
                全称').Value;
          end
          else
          begin
            Application.MessageBox('该号码不存在。','提示',64);
            MaskEdit1.Clear;
            MaskEdit1.SetFocus;
            Exit;
          end;
          StringGrid1.SetFocus;
          StringGrid1.Col := 3;
        end;
    end;
```

下面代码主要实现读取当前单元格坐标，并确定 StringGrid1 是否可以编辑，如果当前行商品编号为空并且商品名称列被选中或者当前行商品编号不为空并且数量列被选中，StringGrid1 可以编辑，否则不可以编辑。

```
    procedure TForm30.StringGrid1SelectCell(Sender: TObject; ACol,
      ARow: Integer; var CanSelect: Boolean);
    begin
      y := ACol;
      x := ARow;
      if (ACol = 3)and( Trim(StringGrid1.Cells[2,x])='')or(ACol =
        4)and(Trim(StringGrid1.Cells[2,x])<>'') then
        StringGrid1.Options := StringGrid1.Options +[goEditing]
      else
        StringGrid1.Options := StringGrid1.Options -[goEditing];
    end;
```

下面代码主要实现在用户录入商品名称时，将商品信息显示在列表中供用户选择，如果用户录入商品数量会统计金额、商品种类、总数量、总金额。

在当前行“商品编号”、“商品名称”、“数量”不为空的情况下，程序根据商品编号查询价格，然后根据数量合计金额。

在用户编辑数据时，如果表格中的数据不为空，程序将统计商品种类、总数量、总金额。

当用户在“商品名称”列中编辑数据时，程序会根据用户输入的数据进行匹配查询，如果有数据返回，会以表格形式显示相关数据，供用户选择。

```
procedure TForm30.StringGrid1SetEditText(Sender: TObject; ACol,
  ARow: Integer; const Value: String);
var
  Jg: Real; //记录价格
  a1: Integer;
begin
  Jg := 0.0;
  if (Trim(StringGrid1.Cells[2,x])<>'')and(Trim(StringGrid1.Cells[3,x])<>'')then
    if (Trim(StringGrid1.Cells[4,x])<>'')and(s2 = False) then
    begin
      with datam.ADOQuery4 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 商品基础信息表 where 商品编号 =:a');
        Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[2,x]);
        Open;
      end;
      Jg:= datam.ADOQuery4.FieldByName('批发价').Value;
      StringGrid1.Cells[5,x]:= FloatToStr(StrToInt(StringGrid1.Cells[4,x])*
          Jg*(StrToFloat(StringGrid1.Cells[6,x])));
      if jcxmlb = True then
      begin
        Label17.Caption := FloatToStr(SumPrice);
        Label21.Caption := IntToStr(SumSL);
        Label15.Caption := IntToStr(Kinds);
      end;
      Exit;
    end
    else
    begin
      StringGrid1.Cells[5,x]:='';
      Exit;
    end;
  if (ss = False)and(s1 = False)then
  begin
    if ACol = 3 then
```

```
        begin
          with datam.ADOQuery1 do
          begin
            Close;
            SQL.Clear;
            SQL.Add('select * from 商品基础信息表 where 商品名称 like :b ');
            Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[3,x])+'%';
            Open;
         end;
          if datam.ADOQuery1.RecordCount>0 then
          begin
            DataSource1.DataSet := datam.ADOQuery1;
            DBGrid1.Visible := True;
          end
          else
          begin
            DataSource1.DataSet := nil;
            DBGrid1.Visible := False;
            S1 := True;
            Application.MessageBox('库中没有此商品。','提示',64 );
            For a1 := 2 to 5 do
              StringGrid1.Cells[a1,x]:= '';
              StringGrid1.SetFocus;
              StringGrid1.Col := 1;
          end;
        end;
      end ;
    end;
```

单击“保存”按钮会将列表中的数据保存到数据库中。在保存时，程序首先检查数据是否为空，如果不为空，向客户集中呼叫票号表中插入数据，同时利用循环向客户集中呼叫表中插入表格中的数据。

```
procedure TForm30.BitBtn4Click(Sender: TObject);
var
  m: Integer;
begin
  //判断数据项是否为空，如果有空项，不允许保存
  if (Trim(Edit4.Text)<>'')and(Trim(Label10.Caption)<>'')and
    (Trim(Label15.Caption)<>'')and(Trim(Label16.Caption)<>'')and(Trim(Label17.Caption)<>
      '')and(JCxmlb = True)then
  begin
    datam.ADOConnection1.BeginTrans;//开始一个事务
    Try
      with datam.ADOQuery2 do
      begin
        Close;//关闭数据集
        SQL.Clear;//清空SQL语句
```

```
            //添加SQL语句
            SQL.Add('insert 客户集中呼叫票号表 Values(:a,:b,:c,:d,Default,:e) ');
            //为参数赋值
            Parameters.ParamByName('a').Value := Trim(Label10.Caption);
            Parameters.ParamByName('b').Value := StrToInt(Label15.Caption);
            Parameters.ParamByName('c').Value := StrToInt(Label16.Caption);
            Parameters.ParamByName('d').Value := StrToFloat(Label17.Caption);
            Parameters.ParamByName('e').Value := Trim(Edit4.Text);
            ExecSQL;//执行SQL语句
          end;
          For m := 1 to StringGrid1.RowCount-1 do
          begin
            with datam.ADOQuery2 do
            begin
              Close;
              SQL.Clear;
              SQL.Add('insert 客户集中呼叫表 Values(:a,:b,:c,:d,:e,:f,:g,:h)');
              Parameters.ParamByName('a').Value := Trim(StringGrid1.Cells[2,m]);
              Parameters.ParamByName('b').Value := Trim(StringGrid1.Cells[3,m]);
              Parameters.ParamByName('c').Value := Trim(StringGrid1.Cells[0,m]);
              Parameters.ParamByName('d').Value := Trim(StringGrid1.Cells[1,m]);
              Parameters.ParamByName('e').Value := StrToInt(StringGrid1.Cells[4,m]);
              Parameters.ParamByName('f').Value := StrToFloat(StringGrid1.Cells[5,m]);
              Parameters.ParamByName('g').Value := StrToFloat(StringGrid1.Cells[6,m]);
              Parameters.ParamByName('h').Value := Trim(Label10.Caption);
              ExecSQL;
            end;
          end;
          datam.ADOConnection1.CommitTrans;//提交事务
          Application.MessageBox('数据保存成功。','提示',64);
          BitBtn7.OnClick(Sender);
        Except
          datam.ADOConnection1.RollbackTrans;//回滚事务
          Application.MessageBox('系统出错。','提示',64);
          Close;
        end;
      end
      else
        Application.MessageBox('项目列表有误,请查看项目列表是否有空项。','提示',64);
    end;
```

21.9.5 客户反馈查询管理程序设计

1. 概述

客户信息反馈查询主要实现客户反馈信息的查询。通过查看客户反馈的信息，用户可以及时地了解市场动态。在该窗口中，用户可以输入或选择客户名称查询客户反馈信息，也可以选择是否使用时间段查询。客户反馈信息查询运行结果如图 21.22 所示。

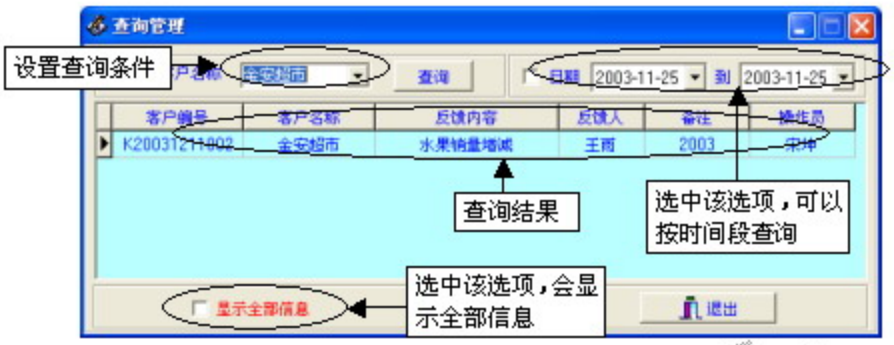


图 21.22　客户反馈信息查询

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TSpeedButton、TComboBox、TCheckBox、TDateTimePicker 等组件。

（2）设置各主要组件属性如表 21.12 所示。

表 21.12　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form1	TForm	Font.Color	clBlue
		Font.Size	8
		BorderStyle	bsDialog
BitBtn	TBitBtn	Kind	bkClose
CheckBox1	TCheckBox	Checked	False
DBGrid1	TDBGrid	DataSource	DataSource1
		Visible	False

3. 程序相关代码

在窗体显示时向 ComboBox1 中添加客户名称，将 CheckBox1 置为非选中状态。

```
procedure TForm1.FormShow(Sender: TObject);
begin
  ComboBox1.Clear;
  DataSource1.DataSet := nil;
  CheckBox1.Checked := False;
  CheckBox1.Checked := False;
  with Datam.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select Distinct 客户名称 from 客户信息反馈表');
    Open;
  end;
  if Datam.ADOQuery1.RecordCount >0 then
  begin
    while Not Datam.ADOQuery1.Eof do
    begin
```

```
        ComboBox1.Items.Add(Datam.ADOQuery1.FieldByName('客户名称').AsString);
        Datam.ADOQuery1.Next;
      end;
    end;
end;
```

下面是“查询”按钮的单击事件代码，单击“查询”按钮会从客户信息反馈表中查询客户反馈信息。程序首先判断查询条件是否为空，如果为空，退出查询操作，否则进一步判断是否按时间段查询（CheckBox2 是否被选中），设置相应的查询语句。

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  if Trim(ComboBox1.Text)<>'' then //是否设置了查询条件
  begin
    if CheckBox2.Checked = False then //不按时间段查询
    begin
      with Datam.ADOQuery2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 客户信息反馈表 where 客户名称 = :a');
        Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
        Open;
      end;
      if Datam.ADOQuery2.RecordCount>0 then
        DataSource1.DataSet := Datam.ADOQuery2
      else
        DataSource1.DataSet := Nil;
    end
    else //按时间段查询
    begin
      with Datam.ADOQuery2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 客户信息反馈表 where 客户名称 = :a and (反馈时间>= :b and 反
            馈时间 <= :c)');
        Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
        Parameters.ParamByName('b').Value := Int(DateTimePicker1.Date);
        Parameters.ParamByName('c').Value := Int(DateTimePicker2.Date)+0.9998;
        Open;
      end;
      if Datam.ADOQuery2.RecordCount>0 then
        DataSource1.DataSet := Datam.ADOQuery2
      else
        DataSource1.DataSet := Nil;
    end;
  end;
```

```
end;
```

用户单击 CheckBox1 时，如果 CheckBox1 被选中，会显示客户信息反馈表中的所有信息。

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked = True then
  begin
    with Datam.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 客户信息反馈表');
      Open;
    end;
    if Datam.ADOQuery2.RecordCount>0 then
      DataSource1.DataSet := Datam.ADOQuery2
    else
      DataSource1.DataSet := Nil;
  end;
end;
```

21.10 疑难问题分析解决

21.10.1 如何将 TDBGrid 中的内容居中显示

在使用 TDBGrid 组件显示数据时，默认情况下，该组件中的数据居左对齐，如何将数据居中对齐呢？

可以在程序设计时双击该组件，打开列表编辑器。单击 Add New 按钮添加数据表字段，选中字段选项，在对象观察器中修改“FieldName”属性为字段名称，设置“AlignMent”属性为“taCenter”，设置“Title.AlignMent”属性为“taCenter”，如图 21.23、图 21.24 所示。



图 21.23 列表编辑器

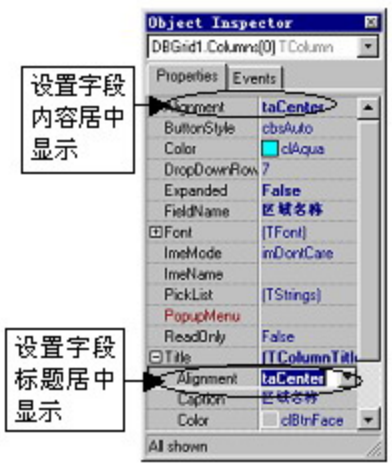


图 21.24 对象观察器

21.10.2 如何更改鼠标指针形状

在 Delphi 中定义的鼠标指针形状共有 22 种，读者可以单击窗口控件的“DragCursor”属性值来查看系统定义的鼠标指针形状。在程序中，可以通过修改“Screen.Cursor”属性值来更改鼠标指针形状。请看下面代码：

```
Screen.Cursors[1]:= LoadCursorFromFile('C:\My Documents\Cur1.cur');
Screen.Cursor := 1;
```

程序首先将光标图像加载到鼠标指针数组中，然后更改当前鼠标指针。需要注意的是，鼠标指针数组的下标值必须在-32768～32767 之间，并且不能为-22～0，因为系统定义的鼠标指针值为-22～0。

21.10.3 如何在控件失去焦点时进行内容检查

为了保证数据的合法性，在很多时候，需要在控件失去焦点时对数据进行合法性检查，如果数据不正确，会进行相应提示。在 Delphi 的窗口控件中提供了 OnExit 事件，表示窗口控件失去焦点时触发该事件，因此，可以在窗口控件的 OnExit 事件中进行数据合法性检查。代码如下：

```
procedure TForm30.Edit8Exit(Sender: TObject);
begin
  if (StrToFloat(Edit8.Text)>1)or(StrToFloat(Edit8.Text)<0)then
  begin
    Application.MessageBox('折扣不能大于1或小于0。','提示',0+64);
    Edit8.Text := '1.0';
    Edit8.SetFocus;
    Exit;
  end;
  try
    StrToFloat(Edit8.Text);
  except
    On EConvertError do
    begin
      Application.MessageBox('请输入合法字符。','提示',0+64);
      Edit8.Text := '1.0';
      Edit8.SetFocus;
      Exit;
    end;
  end;
end;
```

21.10.4 如何防止向数据表中插入重复数据

在开发应用程序时，有时为防止数据表中某些字段重复，会将这些字段设置为主键或为其创建惟一索引。但是，在程序中，如果用户录入了重复的数据，在保存时会出现错误，为防止这些错误的出现，建议在保存前对用户录入的数据进行检查，如果出现重复数据，给予提示，并禁止保存数据。代码如下：

```
with Datam.ADOQuery1 do
```

```
begin
  Close;
  SQL.Clear;
  SQL.Add('select * from 操作员信息表 where 操作员编号= :a');
  Parameters.ParamByName('a').Value   := Trim(Edit1.Text);
  Open;
  end;
  If Datam.ADOQuery1.RecordCount >0 then
  begin
    Application.MessageBox('操作员已经存在，请重新输入。','提示',0+64);
    BitBtn4.OnClick(Sender);
    Exit;
  end;
end;
```

21.10.5 在表格中输入商品数量，如何自动计算金额

商品金额等于商品价格乘以商品数量，相信大家都知道。但是，在字符串网格控件 TStringGrid 中，并没有 OnChange 事件，如何实现商品金额的自动计算呢？在字符串网格控件中提供了 OnSetEditText 事件，该事件在用户编辑单元格中的文本时触发。因此，可以在该事件中计算商品金额。需要注意的是，在计算商品金额前，一定要判断当前行是否有商品，以确定商品价格，防止计算错误。

21.10.6 利用掩码编辑控件格式化字符串

掩码编辑控件（TMaskEdit）提供了限制用户输入的能力，它可以限制用户输入字符个数和类型，并且提供了掩码格式，例如，电话号码、日期格式等。在本章程序的客户呼叫管理模块，作者使用了该控件格式化电话号码。掩码编辑控件包含了一个重要的属性 EditMask。该属性用于限制用户输入的掩码，它由 3 部分组成，中间用分号分隔。第一部分用于确定掩码的格式，所用到的特殊字符如表 21.13 所示；第二部分确定是否将掩码中的格式字符作为数据的一部分，0 表示不作为数据的一部分，1 表示作为数据的一部分；第三部分是指掩码中没有输入数据时出现的替代符。

表 21.13　　掩码编辑控件中的特殊字符

特殊字符	含　义
!	如果“!”出现在掩码中，表示输入字符串的前导空格不会存成数据，如果“!”没有出现在掩码中，表示字符串的末尾空格不会存成字符串
>	如果“>”出现在掩码中，表示其后面的字符都大写，除非遇到“<”
<	如果“<”出现在掩码中，表示其后面的字符都小写，除非遇到“>”
<>	如果“<>”出现在掩码中，表示它后面的所有字符大小写都不会改变
\	“\”表示其后面的字符会作为掩码显示
L	“L”表示在该处只能输入一个字母，并且必须输入一个字母
I	“I”表示在该处只能输入一个字母，但不一定输入
A	“A”表示在该处只能输入字母或数字，要求必须输入
a	“a”表示在该处只能输入字母或数字，但不一定输入

续表

特殊字符	含　义
C	“C” 表示在该处可以输入任何字符，要求必须输入
c	“c” 表示在该处可以输入任何字符，但不一定输入
0	“0” 表示在该处只能输入数字，要求必须输入
9	“9” 表示在该处只能输入数字，但不一定输入
#	“#” 表示在该处只能输入数字或正、负号，但不一定输入
:	“:” 用来分隔时间中的时、分、秒
/	“/” 用来分隔日期中的年、月、日
;	“;” 用来分隔掩码中的3个部分
-	“-” 用来向文本中插入一个空格，在数据输入时，光标会跳过该字符

设置掩码编辑控件的格式，可以双击 EditMask 属性，打开掩码编辑器，各选项设置如图 21.25 所示。

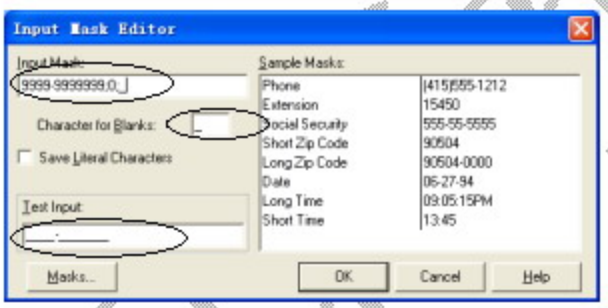


图 21.25　掩码编辑器

21.11　程序调试及错误处理

在开发程序过程中难免会出现各种错误，对于一些语法错误也许会一目了然，很快就能修正错误。但是，对于一些运行时的错误，却很难找到错误根源。为了能够帮助程序开发人员尽快找到并修正错误，Delphi 提供了许多调试工具。在 Delphi 系统菜单“Run”下有一个“Evaluate/Modify”菜单项，单击该菜单项将打开一个对话框，如图 21.26 所示。

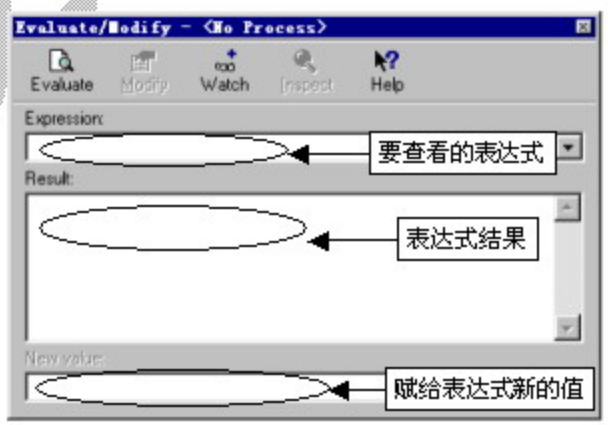


图 21.26　监测与修改对话框

在程序运行时，通过该对话框可以查看和修改变量或对象的属性值。下面将以实例介

绍如何使用该窗口。

以下是一个按钮的单击事件代码：

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: String;
begin
  I:= 's';
  Edit1.Text := i;
  Form1.Caption := 'aa';
  Edit2.Text := i;
  Edit3.Text := 'ssss';
end;
```

（1）在该按钮的单击事件中设置一个断点，效果如图 21.27 所示。

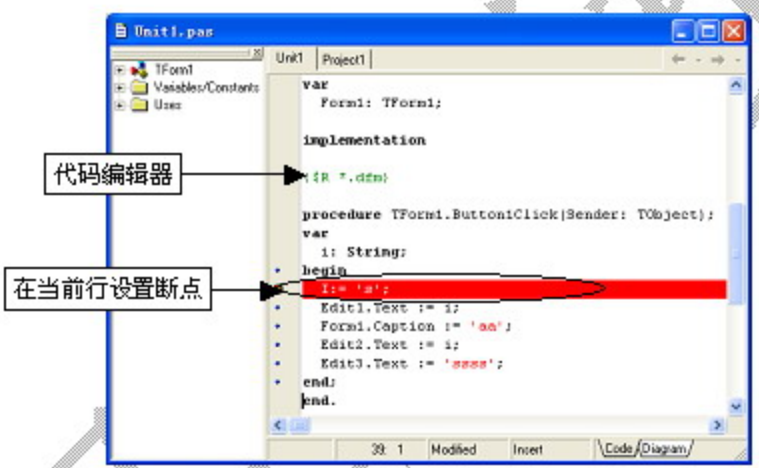


图 21.27　代码编辑器窗口

（2）运行程序，单击按钮，效果如图 21.28 所示。

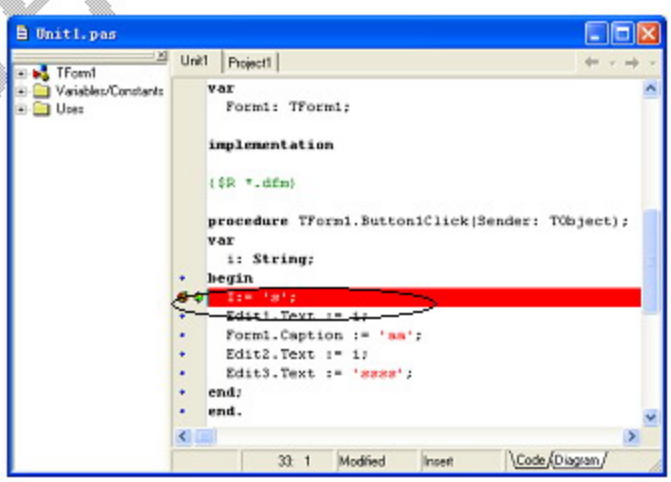


图 21.28　运行期代码窗口

在断点处有一个绿色的箭头，表示程序将要执行的代码，需要注意的是此时当前代码

并没有执行。

（3）按F7键执行当前代码，会发现绿色箭头移到下一条语句上，效果如图 21.29 所示。

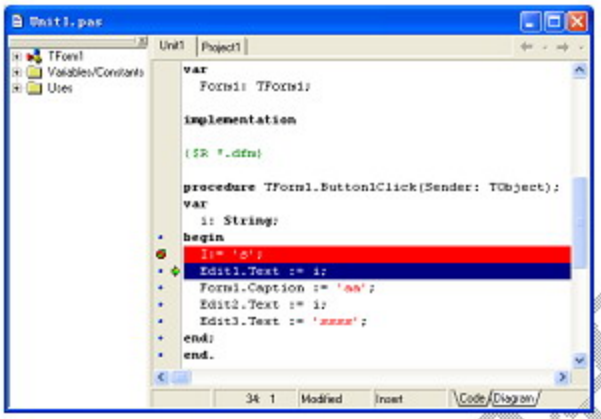


图 21.29　代码编辑窗口

（4）在当前行代码处选中变量“i”，按 Ctrl+F7 键，打开调试窗口，在调试窗口中会发现“i”的值为“s”，如图 21.30 所示。



图 21.30　调试窗口

（5）在代码编辑器中继续按 F7 键，让箭头停留在“Edit2.Text :=i”语句上，然后在调试窗口中修改“i”的值，在“New Value”下拉列表框中设置新值，例如，“明日”，然后单击 Modify 按钮修改数据，效果如图 21.31 所示。



图 21.31　调试窗口

（6）运行程序，效果如图 21.32 所示。

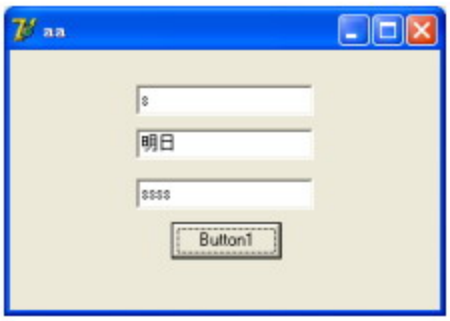


图 21.32　运行结果

21.12　程序设计清单

客户管理系统文件清单如表 21.14 所示。

表 21.14　　客户管理系统设计清单

文　件　名	文 件 类 型	说　　明
Data2.dfm	窗体文件	数据模块
Khhjgl.dfm	窗体文件	客户呼叫管理窗体
Khjbgl.dfm	窗体文件	客户级别管理窗体
Khqygl.dfm	窗体文件	客户区域管理窗体
Khxx1.dfm	窗体文件	显示客户信息窗体
Khxxcx.dfm	窗体文件	客户信息查询窗体
Khxxfkgl.dfm	窗体文件	客户信息反馈管理窗体
Khxxgl.dfm	窗体文件	客户信息管理窗体
Khxxgzgl.dfm	窗体文件	客户信息跟踪管理窗体
Spjcxxgl.dfm	窗体文件	商品基础信息管理窗体
Tjyusc.dfm	窗体文件	操作员管理窗体
Wlzgl.dfm	窗体文件	主窗体
Xtdl2.dfm	窗体文件	系统登录窗体
Yhzhgl.dfm	窗体文件	银行账户管理窗体
Wlgl.dpr	工程文件	
Wlgl.exe	可执行文件	

21.13　技术、经验总结

21.13.1　技术总结

1. 在 TStringGrid 中删除整行数据的实现

在 TStringGrid 中删除整行数据时，如果当前行不是最后一行，不能简单地将 TStringGrid 的行数减一，需要将当前行之后的数据赋给上一行数据，然后清空最后一行数据，如果不清空最后一行数据，当向 TStringGrid 中添加新行时，会将最后一行数据显示在新行中。

2. 鼠标移动时，改变字体颜色

在开发程序时，有时为了增加程序的特色，鼠标在控件上移动时，会改变控件上文本字体的颜色。在 Delphi 的控件类 TControl 中提供了鼠标移动事件“OnMouceMove”，用户可以在该事件中改变控件字体颜色。但是，在鼠标离开控件时，需要恢复控件原来的字体颜色，而 TControl 类并没有提供鼠标离开的事件。仔细想想，当鼠标离开一个控件时，会在另一个控件上移动，因此，会触发另一个控件的“OnMouceMove”事件，可以用该控件的事件恢复字体颜色。代码如下：

```
procedure TForm7.BitBtn1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  Screen.Cursor := -21;
  TBitBtn(Sender).Font.Color := clRed;
end;
//Panel12为BitBtn1的父窗口
procedure TForm7.Panel2MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  Screen.Cursor := -2;
  BitBtn1.Font.Color := clBlue;
  BitBtn2.Font.Color := clBlue;
  BitBtn3.Font.Color := clBlue;
  BitBtn4.Font.Color := clBlue;
  BitBtn5.Font.Color := clBlue;
  BitBtn6.Font.Color := clBlue;
  BitBtn7.Font.Color := clBlue;
  BitBtn8.Font.Color := clBlue;
end;
```

21.13.2 经验总结

通过设计客户管理系统，读者可以了解应用程序的开发步骤。首先了解需求分析，根据需求分析设计系统功能，然后根据功能进行系统分析、系统设计，最后进行编码、程序调试。

程序要完成一组相关的操作，需要使用事务进行控制。例如，在客户呼叫管理系统中，向客户集中呼叫票号表中插入数据后，需要向客户集中呼叫表中插入数据。如果向客户集中呼叫表中插入数据失败，就会使客户信息不完整，此时，需要使用事务进行管理。事务具有原子性，它会使一组操作要么全部执行，要么一个也不执行，这样，如果客户集中呼叫表数据插入失败，客户集中呼叫票号表中也不会有数据插入。

第 22 章 Delphi 在固定资产管理系统中的应用

22.1 固定资产管理系统概述

22.1.1 开发背景及意义

随着社会经济的不断发展，企业规模越来越大，企业资产维护变得越来越困难，尤其是在手工管理方式下，财会人员对资产进行折旧时，需要查阅上一个月的资产信息，根据上月的资产信息核算、折旧资产。如果企业资产很多，可以想象财会人员的工作强度，既要按时对资产进行折旧，即每月对资产进行折旧，又要保证数据的准确性，因为一旦某个月的资产折旧有误，就会影响以后的资产核算。

信息化的发展帮助企业解决了上述难题，计算机的使用使财会人员从繁重的劳动中解脱出来，以往需要对资产信息一项一项核算、折旧，现在只需单击一下按钮，计算机便会自动完成企业所有资产的折旧，而且准确无误。

22.1.2 系统目标

- 在添加和修改资产信息时能够对一些信息进行选择，并且在程序中能够设置这些选项。
- 实现资产信息的添加、修改、删除。
- 实现资产的计提和反计提管理。
- 在程序中对数据库进行备份和恢复。

22.2 系统分析

22.2.1 系统规划

根据上面的系统功能描述，可以定义如下需求分析。

1. 系统目标

利用计算机系统实现企业资产管理。

2. 系统处理范围

- 基础信息管理
- 资产管理
- 系统维护

3. 系统功能

- 资产附加信息管理

在附加信息管理中需要实现资产类别、资产增加方式、资产减少方式、资产使用状况的管理。

- 资产信息管理

在资产信息管理中需要实现资产信息的录入、修改、删除，实现资产的减少、计提、反计提管理。

- 系统维护

在系统维护中需要实现数据的备份和还原。

- 系统初始化

对系统进行初始化。

22.2.2 系统流程图

固定资产管理系统流程图如图 22.1 所示。

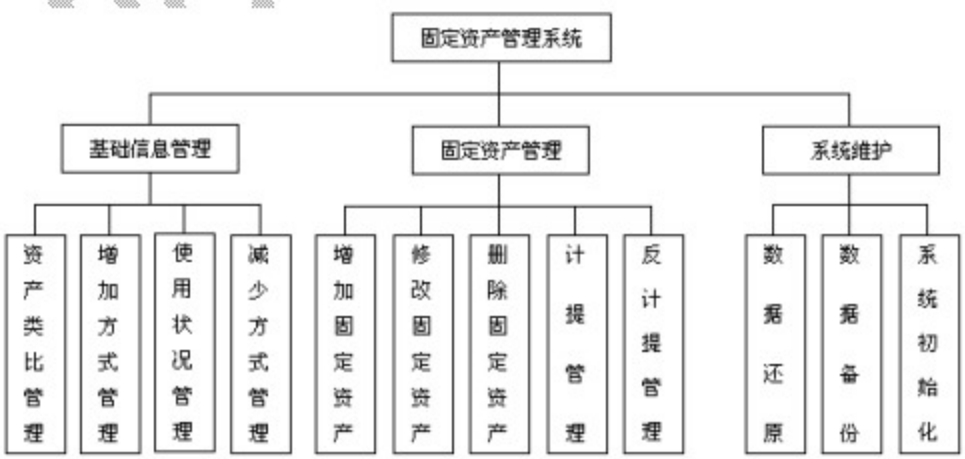


图 22.1 固定资产管理系统流程图

华日科技

22.2.3 业务流程

固定资产管理系统的业务流程图如图 22.2 所示。

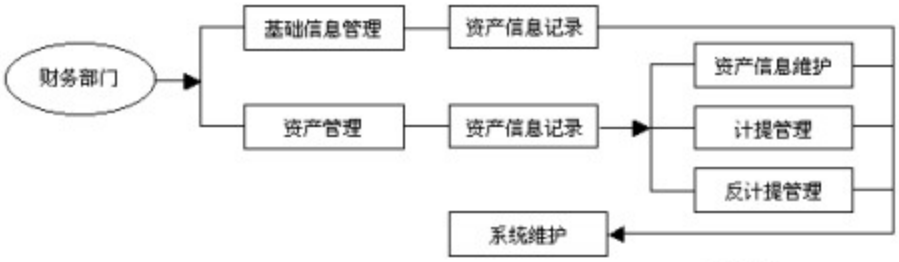


图 22.2 固定资产管理系统业务流程图

22.3 数据库设计

22.3.1 数据库概念设计

固定资产管理系统的 E-R 图如图 22.3 所示。



图 22.3 固定资产管理系统 E-R 图

22.3.2 数据库逻辑结构设计

固定资产管理系统共使用了 7 张数据表，分别为：固定资产类别表、固定资产增加方式表、固定资产减少方式表、固定资产使用状况表、固定资产信息表、计提折旧表、固定资产减少表。由于篇幅关系，在此只列出使用较多的数据表结构(如表 22.1 和表 22.2 所示)，其他表结构请参考光盘。

表 22.1 固定资产信息表

字段名称	字段类型	主键	外键	是否为空	默认值
固定资产编号	Varchar(30)			否	
固定资产名称	Varchar(30)				
固定资产类别	Varchar(30)				
规格型号	Varchar(30)				
增加方式	Varchar(30)				
使用状况	Varchar(30)				
入账原值	Money				
累计折旧	Money				GetDate()
净值	Money				
预计净残值率	Real				

续表

字段名称	字段类型	主键	外键	是否为空	默认值
预计净残值	Money				
折旧方法	Varchar(30)				
预计使用月份	Int				
已计提月份	Int				
月折旧率	Float				
月折旧额	Money				
期初原值	Money				
本年原值调增	Money				
本年原值调减	Money				
年初原值	Money				
期初累计折旧	Money				
本年累计折旧调增	Money				
本年累计折旧调减	Money				
年初累计折旧	Money				
增减	Int				0
入账日期	DateTime				GetDate()

表 22.2　　固定资产减少表

字段名称	字段类型	主键	外键	是否为空	默认值
固定资产编号	Varchar(30)				
固定资产名称	Varchar(30)				
减少方式	Varchar(30)				
清理收入	Money				
清理费用	Money				
清理原因	Varchar(50)				
日期	DateTime				GetDate()

22.3.3　数据字典

数据字典如表 22.3 和表 22.4 所示。

表 22.3　　固定资产信息表

字段名称	描述	说明
固定资产编号	标识企业的一个固定资产	
入账原值	资产在入账时的价值	
累计折旧	资产在入账前的折旧额	
增减	标识资产是否在减少	0 表示增加，1 表示减少

表 22.4　　固定资产减少表

字段名称	描述	说明
固定资产编号	标识企业的一个固定资产	
减少方式	描述资产是如何减少的	
清理原因	记录为何要清理资产	

22.4 处理过程分析

22.4.1 添加固定资产处理过程分析

将资产信息保存到数据表中，首先需要进行登记，由程序产生资产编号，然后录入或选择资产信息。在录入过程中，一些数据（例如：净值）会根据其他数据由程序计算得出。最后是保存数据，将录入的数据保存到数据表中。如图 22.4 所示。

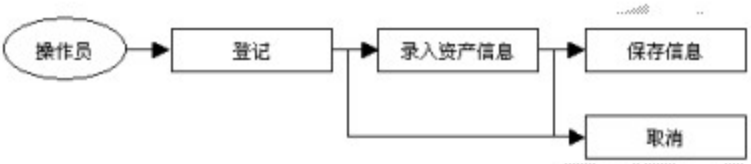


图 22.4 添加固定资产业务流程图

22.4.2 计提折旧处理过程分析

在资产折旧管理中，可以对任意会计期内的固定资产进行折旧，前提是在该会计期内没有进行资产折旧，所以在程序中应允许用户选择会计期。如果用户选择的会计期大于系统当前时间将不允许进行折旧。在进行折旧时，程序会根据当前固定资产的月折旧额修改累计折旧，并将已计提月份加 1。

22.5 主窗体设计

主窗体设计步骤如下。

（1）创建一个窗体，在窗体中放置 TMainMenu、TTreeView、TSplitter、TImageList、TListView、TCoolBar、TToolBar、TSpeedButton、TStatusBar 等组件，如图 22.5 所示。

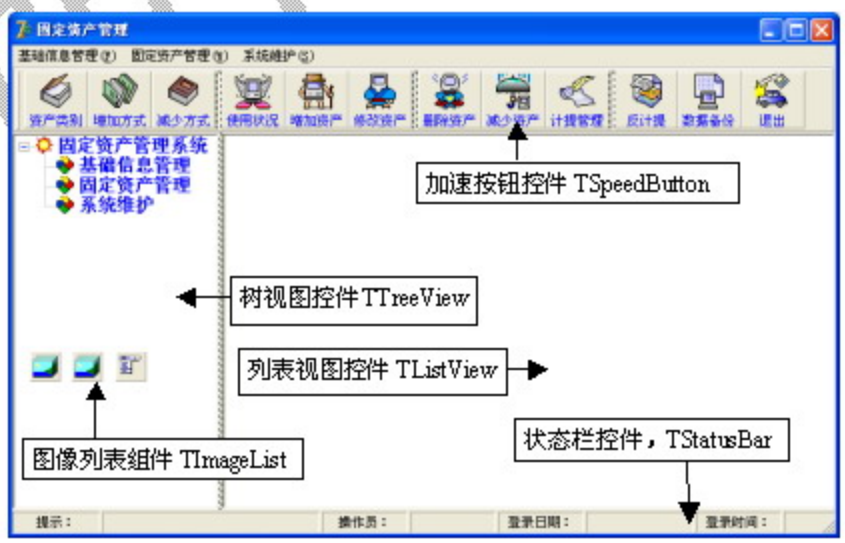


图 22.5 主窗体界面

（2）设置各主要组件属性如表 22.5 所示。

表 22.5　　组件属性设置

对 象 名	对 象 类 型	属　　性	值
Form1	TForm	Font.Color	clBlue
		Font.Size	10
TreeView1	TTreeView	Images	ImageList1
		ShowLines	True
ListView1	TListView	LargeImages	ImageList2

22.6　图像列表设计

图像列表 TImageList 是一组图像的容器，它能够保存大量相同尺寸的图像。包含在 TImageList 中的图像通过索引来访问。下面将介绍如何向图像列表中添加图像。

（1）双击 TImageList 组件打开列表编辑器，如图 22.6 所示。



图 22.6　列表编辑器

（2）单击 Add 按钮，打开图像添加对话框，如图 22.7 所示。

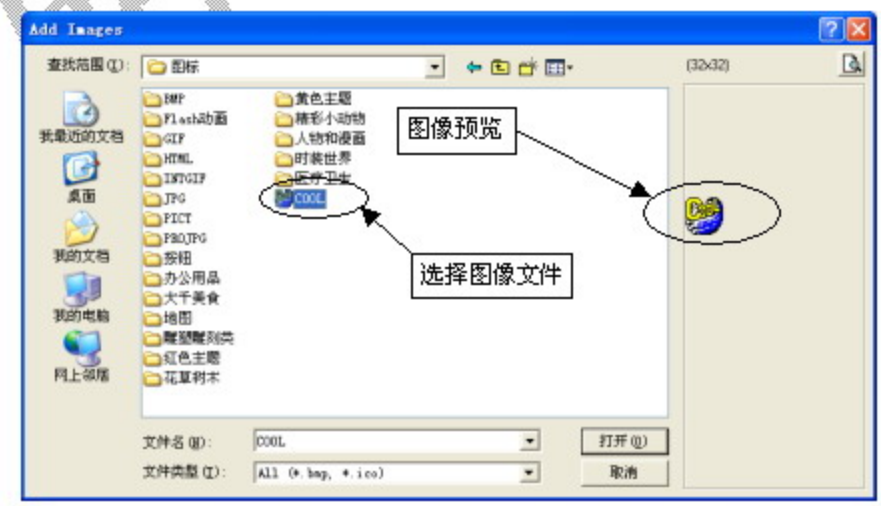


图 22.7　图像添加

（3）选择位图或图标文件，单击“打开”按钮，将图像添加到图像列表中，效果如图22.8所示。



图22.8　图像列表效果图

（4）单击Add按钮可以继续添加图像，单击OK按钮会保存添加的图像。

22.7　数据模块设计

数据模块设计步骤如下。

（1）启动Dephi 7，单击“File”/“New”/“Data Module”菜单项，创建数据模块。

（2）选中数据模块，在对象观察器中修改“Name”属性为“Daz”。

（3）在数据模块上放置TADOConnection组件、TADOQuery组件，如图22.9所示。

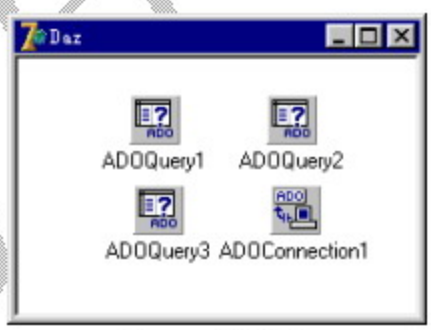


图22.9　数据模块

（4）设置各主要组件属性如表22.6所示。

表22.6　　组件属性设置

对　象　名	对 象 类 型	属　　性	值
ADOConnection1	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=固定资产管理系统
ADOQuery1	TADOQuery	Connection	ADOConnection1
ADOQuery2	TADOQuery	Connection	ADOConnection1
ADOQuery3	TADOQuery	Connection	ADOConnection1

22.8 程序设计与编码

22.8.1 资产类别程序设计

1. 概述

大型企业集团内部固定资产数量众多，各种门类的资产都有可能存在。对于不同类型的资产需要采取不同的管理模式。此外，由于企业的性质、管理方式不同，企业资产分类方法也不尽相同。资产类别管理能够让企业根据自身特点和管理要求确定合理的资产分类方法。资产类别运行结果如图 22.10 所示。



图 22.10 资产类别窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TDataSource、TDBGrid、TEdit 等组件。

（2）设置各主要组件属性如表 22.7 所示。

表 22.7 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form11	TForm	Font.Color	clBlue
		Font.Size	8
DBGrid1	TDBGrid	DataSource	DataSource1
BitBtn3	TBitBtn	Kind	bkClose

3. 程序相关代码

窗体显示时，将资产类别信息显示在界面上。

```
procedure TForm2.FormShow(Sender: TObject);
begin
  Edit1.Clear;//清空编辑框中的内容
  with Daz.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select * from 固定资产类别表');
    Open;
  end;
```

```
    DataSource1.DataSet := Daz.ADOQuery1;
end;
```

下面是“添加”按钮的单击事件代码，程序会将输入的资产类别名称添加到数据表中，同时会刷新数据集。

```
procedure TForm2.BitBtn1Click(Sender: TObject);
begin
  if Trim(Edit1.Text)<>'' then
  begin
    Try
    with Daz.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('Insert  固定资产类别表  values(:a)');
      Parameters.ParamByName('a').Value := Trim(Edit1.Text);
      ExecSQL;
    end;
    Self.OnShow(Sender);//调用窗体的OnShow事件代码
    Application.MessageBox('添加成功。','提示',64);
    Edit1.SetFocus;
    Except
      Application.MessageBox('操作失败。','提示',64);
      Close;
    end;
  end;
end;
```

22.8.2 添加固定资产程序设计

1. 概述

添加固定资产是将固定资产信息保存到数据库中。在添加过程中，固定资产编号由程序自动生成，净值、预计净残值、月折旧率、月折旧额等信息会根据会计电算化中的公式由程序计算得出。当操作员录入完固定资产信息后，单击“保存”按钮会将用户信息保存起来。如果用户在录入过程中，不想添加固定资产信息，可以单击“取消”按钮清除用户录入的数据。添加固定资产运行结果如图 22.11 所示。



图 22.11 增加固定资产窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TComboBox、TPageControl、TDateTimePicker等组件。

（2）设置各主要组件属性如表 22.8 所示。

表 22.8　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form6	TForm	Font.Color	clBlue
		Font.Size	8
DBGrid1	TDBGrid	DataSource	DataSource1
BitBtn4	TbitBtn	Kind	bkClose

3. 程序相关代码

防止在入账原值中输入非法字符。

```
procedure TForm6.Edit3KeyPress(Sender: TObject; var Key: Char);
var
  m: Boolean;
begin
  m := (Key<#8)or(Key>#8)and(Key<#46)or(Key>#46)and(Key<#48)or(Key>#57);
  if m then
    Key := #0;
end;
```

自定义过程 Clear 用于进行界面初始化工作。程序利用“循环遍历”选项卡中的所有控件，如果为编辑框，清空其文本；如果为组合框，将组合框当前文本设置为空。

```
procedure TForm6.Clear;
var
  i: Integer;
begin
  Label3.Caption := '';
  DateTimePicker1.Date := Now();
  For i := 0 to TabSheet1.ControlCount-1 do//利用循环清空编辑框、列表框中的数据
  begin
    if TabSheet1.Controls[i]is TEdit Then
      TEdit(TabSheet1.Controls[i]).Clear
    else if TabSheet1.Controls[i]is TComboBox Then
      TComboBox(TabSheet1.Controls[i]).Text := '';
  end;
  For i := 0 to TabSheet2.ControlCount-1 do
    if TabSheet2.Controls[i]is TEdit Then
      TEdit(TabSheet2.Controls[i]).Clear;
end;
```

下面是 Edit4 的 OnChange 事件，当 Edit4 中的内容改变时会计算相应的数据。Edit3 的 OnChange 事件也调用了该方法。当用户改变“入账原值”和“累计折旧”时，根据用户选择的折旧方法计算月折旧额、净值、预计净残值率。

```
procedure TForm6.Edit4Change(Sender: TObject);
begin
  if Trim(Edit3.Text)<>'' then
  begin
    if (Trim(Edit10.Text)<>'')and(Trim(ComboBox4.Text)='年限平均法(1)') then
      Edit11.Text :=Format('%15.2f',[StrToFloat(Edit3.Text)*StrToFloat(Edit10.Text)/100])
    else if   (Trim(Edit10.Text)<>'')and(Trim(ComboBox4.Text)='年限平均法(2)')then
    begin
      if (Trim(Edit4.Text)<>'')and(Trim(Edit7.Text)<>'')and(Trim(Edit8.Text)<>'')
                                and(Trim(Edit9.Text)<>'')then
        Edit11.Text :=Format('%15.2f',[(StrToFloat(Edit3.Text)-StrToFloat(Edit4.Text)
                    -StrToFloat(Edit7.Text))/(StrToInt(Edit8.Text)-StrToInt(Edit9.Text))])
      else
        Edit11.Clear;
    end
    else
      Edit11.Clear;
  if Trim(Edit4.Text)<>''then
      Edit5.Text := Format('%15.2f',[StrToFloat(Edit3.Text)-StrToFloat(Edit4.Text)])
    else
      Edit5.Clear;
    if Trim(Edit6.Text)<>'' then
      Edit7.Text := Format('%15.2f',[StrToFloat(Edit3.Text)*StrToFloat(Edit6.Text)/100]);
  end
  else
  begin
    Edit5.Clear;
    Edit7.Clear;
    Edit11.Clear;
  end;
end;
```

以下是“保存”按钮的单击事件代码。程序首先检查数据是否为空，数据是否有误（是否出现负数），如果出现上述情况将退出保存操作，否则向固定资产信息表中插入数据。

```
procedure TForm6.BitBtn2Click(Sender: TObject);
begin
  if IsNull = False then
  begin
    if IsWf = False then
    begin
      Try
        with Daz.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('Insert 固定资产信息表
values(:a,:b,:c,:d,:e,:f,:g,:h,:i,:j,:k,:l,:m,:n,:o,:p,:q,:r,:s,:t,:u,:v,:w,:x
```

```
                                                ,Default,:y)');
  Parameters.ParamByName('a').Value := Trim(Label3.Caption);
  Parameters.ParamByName('b').Value := Trim(Edit1.Text);
  Parameters.ParamByName('c').Value := Trim(ComboBox1.Text);
  Parameters.ParamByName('d').Value := Trim(Edit2.Text);
  Parameters.ParamByName('e').Value := Trim(ComboBox2.Text);
  Parameters.ParamByName('f').Value := Trim(ComboBox3.Text);
  Parameters.ParamByName('g').Value := StrToFloat(Edit3.Text);
  Parameters.ParamByName('h').Value := StrToFloat(Edit4.Text);
  Parameters.ParamByName('i').Value := StrToFloat(Edit5.Text);
  Parameters.ParamByName('j').Value := StrToFloat(Edit6.Text);
  Parameters.ParamByName('k').Value := StrToFloat(Edit7.Text);
  Parameters.ParamByName('l').Value := Trim(ComboBox4.Text);
  Parameters.ParamByName('m').Value := StrToInt(Edit8.Text);
  Parameters.ParamByName('n').Value := StrtoInt(Edit9.Text);
  Parameters.ParamByName('o').Value := StrToFloat(Edit10.Text);
  parameters.ParamByName('p').Value := StrToFloat(Edit11.Text);
  if Trim(Edit12.Text)<>'' then
    Parameters.ParamByName('q').Value := StrToFloat(Edit12.Text)
  else
    Parameters.ParamByName('q').Value := Null;
  if Trim(Edit13.Text)<>'' then
    Parameters.ParamByName('r').Value := StrToFloat(Edit13.Text)
  else
    Parameters.ParamByName('r').Value := Null;
  if Trim(Edit14.Text)<>'' then
    Parameters.ParamByName('s').Value := StrToFloat(Edit14.Text)
  else
    Parameters.ParamByName('s').Value := Null;
  if Trim(Edit15.Text)<>'' then
    Parameters.ParamByName('t').Value := StrToFloat(Edit15.Text)
  else
    Parameters.ParamByName('t').Value := Null;
  if Trim(Edit16.Text)<>'' then
    Parameters.ParamByName('u').Value := StrToFloat(Edit16.Text)
  else
    Parameters.ParamByName('u').Value := Null;
  if Trim(Edit17.Text)<>'' then
    Parameters.ParamByName('v').Value := StrToFloat(Edit17.Text)
  else
    Parameters.ParamByName('v').Value := Null;
  if Trim(Edit18.Text)<>'' then
    Parameters.ParamByName('w').Value := StrToFloat(Edit18.Text)
  else
    Parameters.ParamByName('w').Value := Null;
  if Trim(Edit19.Text)<>'' then
    Parameters.ParamByName('x').Value := StrToFloat(Edit19.Text)
```

```
                else
                  Parameters.ParamByName('x').Value := Null;
                  Parameters.ParamByName('y').Value := DateTimepicker1.Date;
                ExecSQL;
              end;
              Application.MessageBox('操作成功。','提示',64);
              BitBtn3.OnClick(Sender);
            Except
              Application.MessageBox('操作失败。','提示',64);
              Close;
            end;
          end
          else
            Application.MessageBox('项目列表数据不能小于零。','提示',64);
        end
        else
          Application.MessageBox('基础信息有空项。','提示',64);
      end;
```

22.8.3 修改固定资产程序设计

1. 概述

修改固定资产是对已保存到数据表中的固定资产信息进行部分修改。操作员在添加固定资产时，由于原始数据不准确或操作员的录入错误，导致保存到数据表中的固定资产信息不准确。此时，操作员需要通过“修改固定资产”窗口修改固定资产信息。“修改固定资产”窗口运行结果如图 22.12 所示。

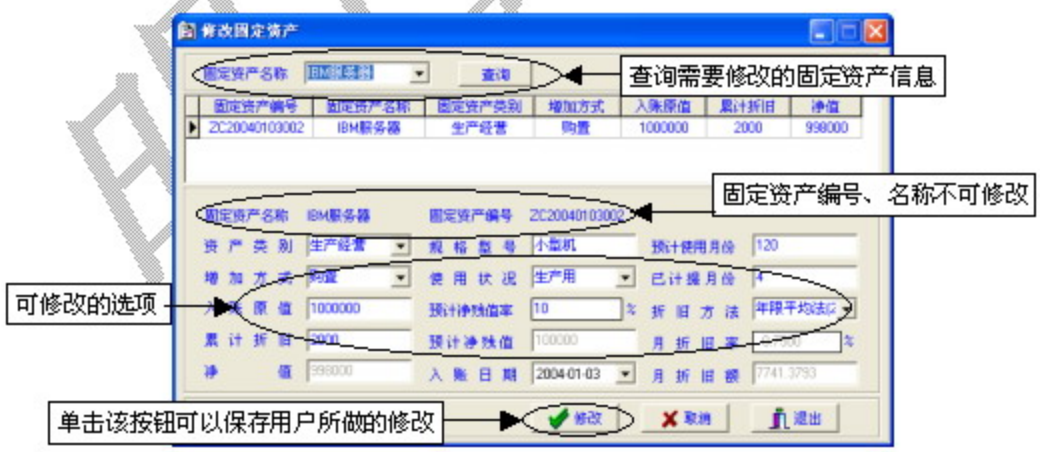


图 22.12 “修改固定资产”窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TSpeedButton、TLabel、TEdit、TComboBox、TDataSource、TDBGrid、TDataTimePicker 等组件。

（2）设置各主要组件属性如表 22.9 所示。

表 22.9　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form9	TForm	Font.Color	clBlue
		Font.Size	8
DBGrid1	TDBGrid	DataSource	DataSource1
BitBtn1	TBitBtn	Enabled	False

3. 程序相关代码

下面是“查询”按钮的单击事件代码。程序判断用户是否设置了查询条件，如果没有设置则退出查询，否则根据条件查询固定资产信息，如果有数据返回，显示数据，并将“修改”按钮置为可用；没有数据返回，将“修改”按钮置为不可用。

```
procedure TForm9.SpeedButton1Click(Sender: TObject);
begin
  if Trim(ComboBox1.Text)<>'' then//是否设置了查询条件
  begin
    with Daz.ADOQuery2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 固定资产信息表 where 固定资产名称 = :a');
      Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
      Open;
    end;
    if Daz.ADOQuery2.RecordCount >0 then //是否有数据返回
    begin
      DataSource1.DataSet := Daz.ADOQuery2;
      EditValue;
      BitBtn1.Enabled := True;
    end
    else
    begin
      DataSource1.DataSet := Nil;
      Label19.Caption := '';
      Label20.Caption := '';
      BitBtn1.Enabled := False;
    end;
  end;
end;
```

自定义过程 EditValue 将数据集当前记录数据显示在界面上。

```
procedure TForm9.EditValue;
begin
  Label20.Caption := Daz.ADOQuery2.FieldByName('固定资产编号').Value;
  Label19.Caption := Daz.ADOQuery2.FieldByName('固定资产名称').Value;
  ComboBox2.Text :=   Daz.ADOQuery2.FieldByName('固定资产类别').Value;
```

```
    ComboBox3.Text :=   Daz.ADOQuery2.FieldByName('增加方式').Value;
    Edit3.Text := FloatToStr(Daz.ADOQuery2.FieldByName('入账原值').Value);
    Edit4.Text := FloatToStr(Daz.ADOQuery2.FieldByName('累计折旧').Value);
    Edit5.Text := FloatToStr(Daz.ADOQuery2.FieldByName('净值').Value);
    Edit2.Text := Daz.ADOQuery2.FieldByName('规格型号').Value;
    ComboBox4.Text := Daz.ADOQuery2.FieldByName('使用状况').Value;
    Edit6.Text := FloatToStr(Daz.ADOQuery2.FieldByName('预计净残值率').Value);
    Edit7.Text := FloatToStr(Daz.ADOQuery2.FieldByName('预计净残值').Value);
    DateTimePicker1.Date := Daz.ADOQuery2.FieldByName('入账日期').Value;
    Edit8.Text := IntToStr(Daz.ADOQuery2.FieldByName('预计使用月份').Value);
    Edit9.Text := IntToStr(Daz.ADOQuery2.FieldByName('已计提月份').Value);
    ComboBox5.Text := Daz.ADOQuery2.FieldByName('折旧方法').Value;
    Edit10.Text := Format('%8.4f',[Daz.ADOQuery2.FieldByName('月折旧率').AsFloat]);
    Edit11.Text := FloatToStr(Daz.ADOQuery2.FieldByName('月折旧额').Value);
end;
```

以下是“修改”按钮的单击事件代码。程序首先提示是否修改数据，如果选择“是”，则进一步检查数据是否有误（固定资产数据是否为空，是否出现负数），如果有误，进行提示，并退出修改操作，反之修改数据。

```
procedure TForm9.BitBtn1Click(Sender: TObject);
begin
  if Application.MessageBox('确实要修改原始数据吗？','提示',MB_YesNo)=ID_Yes then
  begin
    if IsNull = False then
    begin
      if IsWf = False then
      begin
        Try
          With Daz.ADOQuery2 do
          begin
            Edit;
            FieldByName('固定资产编号').Value := Trim(Label20.Caption);
            FieldByName('固定资产名称').Value := Trim(Label19.Caption);
            FieldByName('固定资产类别').Value := Trim(ComboBox2.Text);
            FieldByName('增加方式').Value := Trim(ComboBox3.Text);
            FieldByName('入账原值').Value := StrToFloat(Edit3.Text);
            FieldByName('累计折旧').Value := StrToFloat(Edit4.Text);
            FieldByName('净值').Value := StrToFloat(Edit5.Text);
            FieldByName('规格型号').Value := Trim(Edit2.Text);
            FieldByName('使用状况').Value := Trim(ComboBox4.Text);
            FieldByName('预计净残值率').Value := StrToFloat(Edit6.Text);
            FieldByName('预计净残值').Value := StrToFloat(Edit7.Text);
            FieldByName('入账日期').Value := DateTimePicker1.Date;
            FieldByName('预计使用月份').Value := StrToInt(Edit8.Text);
            FieldByName('已计提月份').Value := StrToInt(Edit9.Text);
            FieldByName('折旧方法').Value := Trim(ComboBox5.Text);
            FieldByName('月折旧率').Value := StrToFloat(Edit10.Text);
```

```
                FieldByName('月折旧额').Value := StrToFloat(Edit11.Text);
                Post;
              end;
              BitBtn2.OnClick(Sender);
              BitBtn1.Enabled := False;
              Application.MessageBox('修改成功。','提示',64);
            Except
              Application.MessageBox('修改失败。','提示',64);
              Close;
            end;
          end
          else
            Application.MessageBox('基础数据不能小于零。','提示',64);
        end
        else
          Application.MessageBox('基础数据不能为空。','提示',64);
    end;
end;
```

函数 IsNull 判断数据是否为空，如果为空，函数返回值为 True，否则函数返回值为 False。

```
function TForm9.IsNull: Boolean;
begin
   IsNull := False;
   if (Trim(label19.Caption)='')or(Trim(Label20.Caption)='')or(Trim(ComboBox2.Text)='')
or(Trim(ComboBox3.Text)='')or(Trim(Edit3.Text)='')or(Trim(Edit4.Text)='')or
(Trim(Edit5.Text)='')or(Trim(ComboBox4.Text)='')or(Trim(Edit6.Text)='')
or(Trim(Edit7.Text)='')or(Trim(Edit8.Text)='')or(Trim(Edit9.Text)='')or
(Trim(ComboBox5.Text)='')or(Trim(Edit10.Text)='')or(Trim(Edit11.Text)='') then
       IsNull := True;
end;
```

22.8.4 计提折旧程序设计

1. 概述

计提折旧是固定资产管理系统的核心，它能够实现自动计提本期固定资产折旧。本系统的计提折旧对本会计期添加的固定资产不计提折旧，下期计提折旧；对本期减少的固定资产本期继续计提，下期停止计提折旧。计提折旧运行结果如图 22.13 所示。

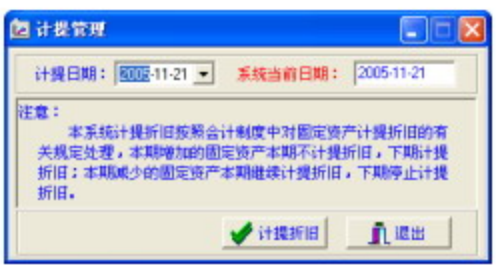


图 22.13 计提折旧运行窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TMemo、TDateTimePicker 等组件。

（2）设置各主要组件属性如表 22.10 所示。

表 22.10　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form8	TForm	Font.Color	clBlue
		Font.Size	8
BitBtn2	TBitBtn	Kind	bkClose

3. 程序相关代码

以下是“计提折旧”按钮的单击事件代码，程序会根据用户选择的会计期进行相应的折旧。

固定资产的折旧较为复杂，程序首先检查折旧时间是否合法，然后确定本期是否进行了折旧，如果进行了折旧，将不允许再折旧，最后进行折旧操作。在折旧时，首先查询本期增加的固定资产信息和已减少的固定资产信息之外的信息，因为本期增加的固定资产，本期不进行折旧，下期进行折旧。利用循环遍历每条记录，检查预计使用月份是否小于已计提月份，如果小于，表明固定资产已过期，不折旧该资产，否则进行折旧，修改已计提月份（加 1），根据固定资产的折旧方法计算修改累计折旧、月折旧额。本期减少的固定资产，本期继续计提折旧，所以还需要查询本期减少的固定资产，然后进行折旧，方法与前面讲述的相同。

```
procedure TForm8.BitBtn1Click(Sender: TObject);
var
  bj: String;//计提标记
  year,month,day: Word;
  i: Integer;
begin
  mm := False;//mm为全局变量，当本期有可折旧的固定资产时为True
  nn := False;//nn为全局变量，当本期有减少的固定资产时为True
  if Trunc(aa) > Trunc(DateTimePicker1.Date) then
  begin
    Application.MessageBox('计提折旧时间不能小于固定资产添加时间。','提示',64);
    DateTimePicker1.Date := Now();
    Exit;
  end;
  if Trunc(DateTimePicker1.Date)> Trunc(Now()) then
  begin
     Application.MessageBox('计提折旧时间不能大于系统当前时间。','提示',64);
     Exit;
  end;
  DecodeDate(DateTimePicker1.Date,year,month,day);
  bj:= FormatDateTime('yyyymm',DateTimePicker1.Date);
  with Daz.ADOQuery1 do //判断本期是否进行了计提折旧，如果已经折旧，不允许再折旧
  begin
     Close;
     SQL.Clear;
     SQL.Add('select * from 计提折旧表 where 计提标记 = :a');
     Parameters.ParamByName('a').Value := Trim(bj);
     Open;
  end;
```

```
if Daz.ADOQuery1.RecordCount >0 then
begin
   Application.MessageBox('本期固定资产已经计提了折旧费用。','提示',64);
   Exit;
end;
Try
Daz.ADOConnection1.BeginTrans;
with Daz.ADOQuery1 do //查询本期可折旧的固定资产
begin
   Close;
   SQL.Clear;
   SQL.Add('select * from 固定资产信息表 where 增减 = 0 and 固定资产编号 not in ( select
            固定资产编号 from 固定资产信息表 where Year(入账日期)= :a and Month(入账日期)= :b)
            and 增减 = 0 ');
   Parameters.ParamByName('a').Value := Year;
   Parameters.ParamByName('b').Value := Month;
   Open;
end;
if Daz.ADOQuery1.RecordCount>0 then
begin
   mm := True; //本期有可折旧的固定资产
   Daz.ADOQuery1.Edit;
   while Not Daz.ADOQuery1.Eof do
   begin
      if Daz.ADOQuery1.FieldByName('预计使用月份').Value >Daz.ADOQuery1.FieldByName('已计
                        提月份').Value then
      begin
         Daz.ADOQuery1.Edit;
         if Trim(Daz.ADOQuery1.FieldByName('折旧方法').Value) = '年限平均法(1)' then
         begin
            Daz.ADOQuery1.FieldByName('累计折旧').Value :=Daz.ADOQuery1.FieldByName('累计折
                        旧').Value + Daz.ADOQuery1.FieldByName('月折旧额').Value;
            Daz.ADOQuery1.FieldByName('已计提月份').Value := Daz.ADOQuery1.FieldByName('已
                        计提月份').Value +1;
            Daz.ADOQuery1.Post;
         end
         else if Trim(Daz.ADOQuery1.FieldByName('折旧方法').Value) = '年限平均法(2)' then
         begin
            with Daz.ADOQuery1 do
            begin
               FieldByName('已计提月份').Value := FieldByName('已计提月份').Value +1;
               FieldByName('月折旧额').Value := (FieldByName('入账原值').Value –
                        FieldByName('累计折旧').Value - FieldByName('预计净残值').Value)
                        /(FieldByName('预计使用月份').Value-FieldByName('已计提月份').Value);
               Post;
            end;
         end;
```

```
        end;
        Daz.ADOQuery1.Next;
      end;
    end
    else
      mm := False;
    with Daz.ADOQuery1 do //如果本期有减少的固定资产，本期继续折旧
    begin
      Close;
      SQL.Clear;
      SQL.Add('select * from 固定资产信息表 where 固定资产编号 in (Select 固定资产编号 From
                固定资产减少表 where year(日期)=:a and Month(日期) = :b)');
      If Month = 1 then
      begin
        Parameters.ParamByName('a').Value := Year -1;
        Parameters.ParamByName('b').Value := 12;
      end
      else
      begin
        Parameters.ParamByName('a').Value := Year;
        Parameters.ParamByName('b').Value := Month -1;
      end;
      Open;
    end;
    if Daz.ADOQuery1.RecordCount>0 then
    begin
      nn := True;
      Daz.ADOQuery1.Edit;
      while Not Daz.ADOQuery1.Eof do
      begin
        if Daz.ADOQuery1.FieldByName('预计使用月份').Value >Daz.ADOQuery1.FieldByName('已计
                    提月份').Value then
        begin
          if Trim(Daz.ADOQuery1.FieldByName('折旧方法').Value) = '年限平均法(1)' then
          begin
            Daz.ADOQuery1.FieldByName('累计折旧').Value :=Daz.ADOQuery1.FieldByName('累计折
                    旧').Value + Daz.ADOQuery1.FieldByName('月折旧额').Value;
            Daz.ADOQuery1.FieldByName('已计提月份').Value := Daz.ADOQuery1.FieldByName('已
                    计提月份').Value +1;
            Daz.ADOQuery1.Post;
          end
          else if Trim(Daz.ADOQuery1.FieldByName('折旧方法').Value) = '年限平均法(2)' then
          begin
            with Daz.ADOQuery1 do
            begin
              FieldByName('已计提月份').Value := FieldByName('已计提月份').Value +1;
              FieldByName('月折旧额').Value := (FieldByName('入账原值').Value -FieldByName('
```

```
                累计折旧').Value - FieldByName('预计净残值').Value)/(FieldByName('预计使用月
                份').Value-FieldByName('已计提月份').Value);
              Post;
            end;
          end;
        end;
        Daz.ADOQuery1.Next;
      end;
    end
    else
      nn := False;
    if (mm = True)or(nn = True) then
    begin
      with Daz.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('insert 计提折旧表 values(:a,:b)');
        Parameters.ParamByName('a').Value := Trim(BJ);
        Parameters.ParamByName('b').Value := DateTimePicker1.Date;
        ExecSQL
      end;
    end ;
    Daz.ADOConnection1.CommitTrans;
    if (mm = False)and(nn = False) then
      Application.MessageBox('本期没有可计提折旧的固定资产，固定资产可能本期刚添加或已经减
                少。','提示',64)
    else Application.MessageBox('计提折旧成功。','提示',64);
    Except
      Daz.ADOConnection1.RollbackTrans;
      Application.MessageBox('计提折旧失败。','提示',64);
    end;
  end;
```

22.8.5 数据备份程序设计

1. 概述

数据备份是对程序中所使用的数据库进行备份。在计算机中，由于病毒的入侵或磁盘的损坏，经常会导致数据的丢失。如果在程序中能对数据库进行备份，当系统遭到破坏时，便可以使用备份文件进行恢复。数据备份运行结果如图 22.14 所示。

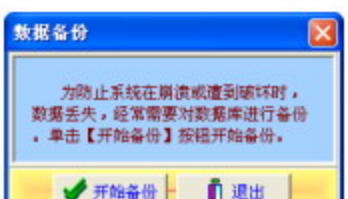


图 22.14 数据备份窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TMemo、TChart、TBitBtn 等组件。

（2）设置各主要组件属性如表 22.11 所示。

表 22.11　　组件属性设置

对 象 名	对 象 类 型	属　性	值
FormF	TForm	Font.Color	clBlue
		Font.Size	8
BitBtn2	TBitBtn	Kind	bkClose

3. 程序相关代码

以下是“开始备份”按钮的单击事件代码，程序利用 SQL 语句进行数据库备份。

```
procedure TFormF.BitBtn1Click(Sender: TObject);
begin
  Try
  with Daz.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('Backup database  固定资产管理系统  to Disk = "C:\gdzcback.bak" ');
    ExecSQL;
  end;
  Application.MessageBox('数据备份成功。','提示',0+64);
  Except
  Application.MessageBox('系统出错。','提示',0+64);
  Close;
  end;
end;
```

22.8.6　系统初始化程序设计

1. 概述

当用户首次使用数据库应用程序时，需要对数据表进行初始化，即清空数据表中的数据。系统初始化为用户提供了初始化数据表的功能。实现系统初始化可以有多种方法，比较简单的方法是创建一个存储过程，在存储过程中编写 SQL 语句删除数据表中的数据。还有一种更为简单的方法，即执行数据库脚本，利用数据库脚本删除并重建数据库，这样，数据库中原来的数据便会丢失。采用第二种方法，需要先编写数据库脚本，然后在程序中调用查询分析器执行脚本。在本章程序中，作者采用了第一种方法，即利用自定义的存储过程删除数据表中的数据。系统初始化运行结果如图 22.15 所示。



图 22.15　系统初始化窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TSpeedButton、TImage、TLabel 等组件。

（2）设置各主要组件属性如表 22.12 所示。

表 22.12　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form69	TForm	Font.Color	clBlue
		Font.Size	8
SpeedButton1	TSpeedButton	Caption	删除

3. 程序相关代码

以下是“删除”按钮的单击事件代码。在进行初始化时会进行提示，并检查是否存在备份文件，如果不存在备份文件，不允许初始化。在进行初始化操作时，系统调用存储过程删除数据表中的数据。

```
procedure TForm69.SpeedButton1Click(Sender: TObject);
begin
  if Application.MessageBox('确实要删除全部数据吗？','提示',MB_YesNo)= ID_yes then
  begin
    Try
      if FileExists('C:\gdzcBack.Bak') then
      begin
        with Daz.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          //DeleteTable 是预先创建的存储过程
          SQL.Add('exec DeleteTable');
          ExecSQL;
        end;
        Application.MessageBox('系统初始化成功。','提示',0+64);
      end
      else
        Application.MessageBox('数据库没有备份，不能初始化数据，请先备份数据库。','提示
',64);
    Except
      Application.MessageBox('系统出错。','提示',0+64);
      Close;
    end;
  end;
end;
```

22.9　疑难问题分析解决

22.9.1　如何创建存储过程

在本系统的系统初始化模块中使用了存储过程“DeleteTable”实现数据的初始化。创建存储过程“DeleteTable”如图 22.16 所示。

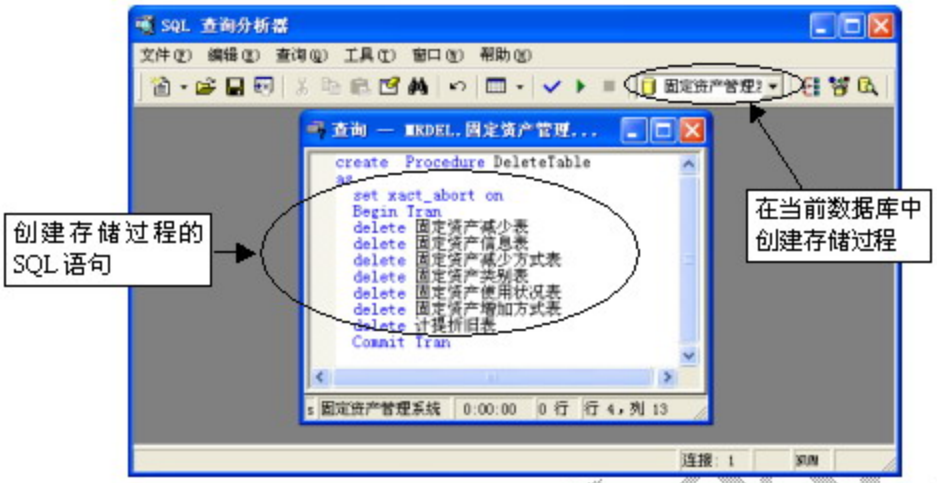


图 22.16　创建存储过程

“Create Procedure”是创建存储过程的关键字，“DeleteTable”为要创建的存储过程名称。“as”表示下面是存储过程要执行的 SQL 语句。“set xact_abort on”表示当指定的 SQL 语句运行时产生错误，SQL Server 将自动回滚当前事务。“Begin Tran”表示开始一个事务，“Commit Tran”表示提交事务。

22.9.2　数据还原的实现

在 SQL Server 2000 中，还原数据库使用“Restore DataBase”关键字。下面是还原“固定资产管理系统”数据库的 SQL 语句。

```
use Master Restore DataBase 固定资产管理系统 From Disk = "C:\gdzcback.bak"
```

“C:\gdzcback.bak”是“固定资产管理系统”数据库备份文件。

“use Master”是将当前数据集组件连接到“Master”（SQL Server 2000 自带的系统数据库）数据库上。如果不写“use Master”语句，在还原数据库时会出错。因为在 SQL Server 2000 中，如果有应用程序在访问数据库，将不允许还原该数据库，直到没有应用程序访问该数据库。

22.9.3　查看 SQL Server 的备份文件

在本章程序的数据还原模块中，程序会查看数据库备份文件的内容，以确定某一天是否进行了备份，如果当天有多个备份，会以列表形式显示具体的备份时间，以便用户将数据还原到某一时刻。在 SQL Server 2000 中提供了“Restore HeaderOnly”语句查看备份文件信息，语法格式如下：

```
RESTORE HEADERONLY
FROM < backup_device >
```

参数说明：

RESTORE HEADERONLY 是关键字，表示查看备份文件信息。

From 关键字表示从哪个备份设备上查看备份信息。

backup_device 表示备份设备。

示例如图 22.17 所示。

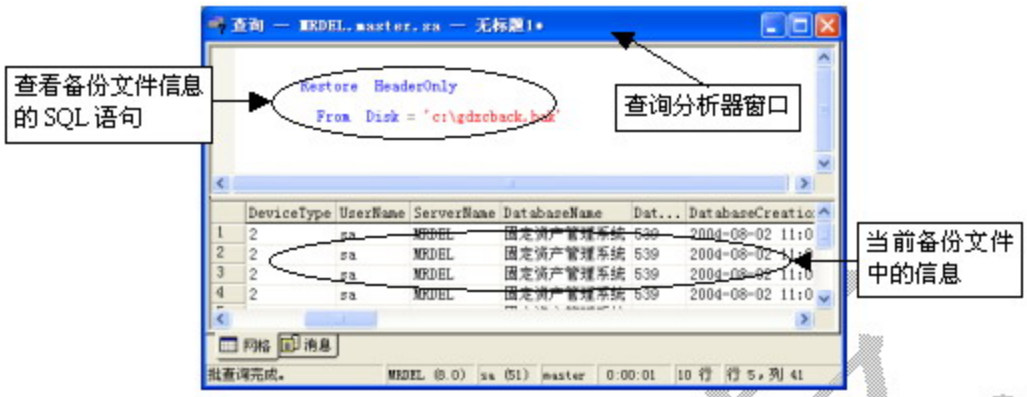


图 22.17　查看备份文件信息

22.9.4　利用循环语句清空界面数据

在开发程序过程中，经常需要清空界面中的数据。例如，在完成一次用户录入操作后，需要清空用户录入的数据，以方便用户下一次录入数据。如果为每一个编辑框或组合框编写一条语句，可能会很繁琐。其实，完全可以使用循环语句来清空控件中的文本。代码如下：

```
procedure TForm9.Clear;
var
  i: Integer;
begin
  //Panel3为编辑框或组合框的容器
  For i := 0 to Panel3.ControlCount-1 do
  begin
    if Panel3.Controls[i]is TEdit Then
      TEdit(Panel3.Controls[i]).Clear
    else if Panel3.Controls[i]is TComboBox Then
      TComboBox(Panel3.Controls[i]).Text := '';
  end;
end;
```

22.9.5　固定资产基本概念

❑　入账原值

入账原值是固定资产取得时根据有关计价原则确定的原始成本。

❑　累计折旧

累计折旧是指核算企业账面固定资产的计提折旧数额，用于反映固定资产由于有形损耗或无形损耗而减少的价值。

❑　净值

净值是入账原值与累计折旧的差额。

❑　预计净残值

预计净残值是固定资产不能再投入使用，在报废清理时，其剩余残料或零件的残余价

值。由于预计净残值不能准确预测，因此，财务制度规定商品流通企业固定资产的预计残值一般为其原值的 3%~5%。

- 预计净残值率

预计净残值率是预计净残值占入账原值的百分比。

22.9.6 固定资产计算公式

固定资产的核算方法有多种，例如，平均年限法、工作量法、加速折旧法、双倍余额递减法、年数总和法等。由于篇幅关系，不能在此做一一介绍，下面只以平均年限法为例介绍固定资产的核算公式，其他核算方法请读者参阅其他相关书籍。

- 月折旧率

月折旧率=(1 – 净残值率) / 使用年限

- 月折旧额

月折旧额=入账原值×月折旧率

- 净值

净值=入账原值-累计折旧

- 预计净残值

预计净残值=入账原值×净残值率

22.10 程序测试

在完成了一个应用程序的开发之后，便进入系统测试阶段。系统测试将检查应用程序是否完成了用户的需求，软件在操作过程中是否存在漏洞。下面将以测试添加固定资产管理为例介绍测试程序的一般方法。

1. 测试程序是否完成预定的功能

（1）运行程序，打开固定资产录入窗口，单击“添加”按钮，查看程序是否生成了固定资产编号。

（2）录入固定资产名称、入账原值等信息，查看净值、预计净残值、月折旧率、月折旧额等数据是否由程序计算得出。

（3）单击“保存”按钮查看数据是否真的保存到数据表中。

2. 测试基础信息是否完整

在设计基础信息时，最基本的功能是保证所有信息均存储到数据表中。在进行保存操作时，应检查用户输入的数据是否完整，如果不完整应进行提示，并退出操作，以防止由于用户疏忽而导致的错误。

（1）运行程序，打开固定资产录入窗口，单击“保存”按钮，查看程序是否出错，会发现程序出现如图 22.18 所示的提示信息。



图 22.18 提示信息

（2）检查“保存”按钮的程序代码，发现程序是通过自定义函数 IsNull 判断基础信息是否为空的。

清华科技

```
procedure TForm6.BitBtn2Click(Sender: TObject);
begin
  if IsNull = False then //自定义函数，判断用户录入的数据是否为空
  ……
end
function TForm6.IsNull: Boolean;
begin
  IsNull := False;
  if (Trim(label3.Caption)='')or(Trim(Edit1.Text)='')or(Trim(ComboBox1.Text)='')
  or(Trim(ComboBox2.Text)='')or(Trim(Edit3.Text)='')or(Trim(Edit4.Text)='')or
  (Trim(Edit5.Text)='')or(Trim(ComboBox3.Text)='')or(Trim(Edit6.Text)='')or
  (Trim(Edit7.Text)='')or(Trim(Edit8.Text)='')or(Trim(Edit9.Text)='')or
  (Trim(ComboBox4.Text)='')or(Trim(Edit10.Text)='')or(Trim(Edit11.Text)='') then
      IsNull := True;
end;
```

22.11 程序设计清单

固定资产管理系统文件清单如表 22.13 所示。

表 22.13　　固定资产设计清单

文 件 名	文 件 类 型	说　明
Bfsb.dfm	窗体文件	显示备份时间窗体
Dataz.dfm	窗体文件	数据模块
Fjtzj.dfm	窗体文件	反计提折旧窗体
Gdzcgl.dfm	窗体文件	主窗体
Gdzcjsgl.dfm	窗体文件	固定资产减少窗体
Gdzclr.dfm	窗体文件	固定资产录入窗体
Jtzj.dfm	窗体文件	计提折旧窗体
Scgdzc.dfm	窗体文件	册除固定资产窗体
Sjbf.dfm	窗体文件	数据备份窗体
Sjhy.dfm	窗体文件	数据还原窗体
Xggdzc.dfm	窗体文件	修改固定资产窗体
Xtcsh.dfm	窗体文件	系统初始化窗体
Zcjsgl.dfm	窗体文件	资产减少方式窗体
Zclbgl.dfm	窗体文件	资产类别管理窗体
Zcsyqkgl.dfm	窗体文件	资产使用情况管理
Zczjfsgl.dfm	窗体文件	资产增加方式窗体
Zcgl.dpr	工程文件	
Zcgl.exe	可执行文件	

22.12 技术、经验总结

22.12.1 技术总结

在 Delphi 中，当打开一个新的工程文件时，当前工程就会关闭。假如某个项目需要使用其他项目的内容，浏览起来很不方便。此时，可以创建一个项目组，将某些有关联的项目放到一个项目组中，这样，在项目组中就可以同时浏览所有的项目文件。下面介绍创建项目组的方法。

（1）单击“View”/“Project Manager”菜单项，打开项目管理器，如图 22.19 所示。

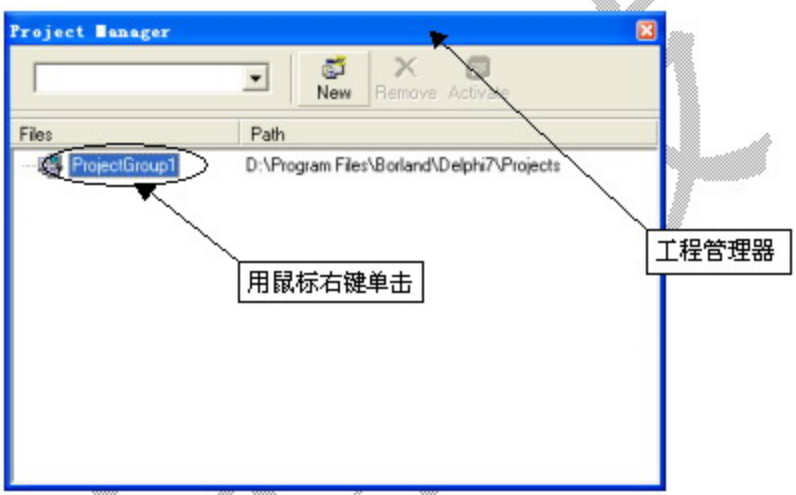


图 22.19 项目管理器

（2）用鼠标右键单击“ProjectGroup1”选项，将弹出一个快捷菜单，如图 22.20 所示。

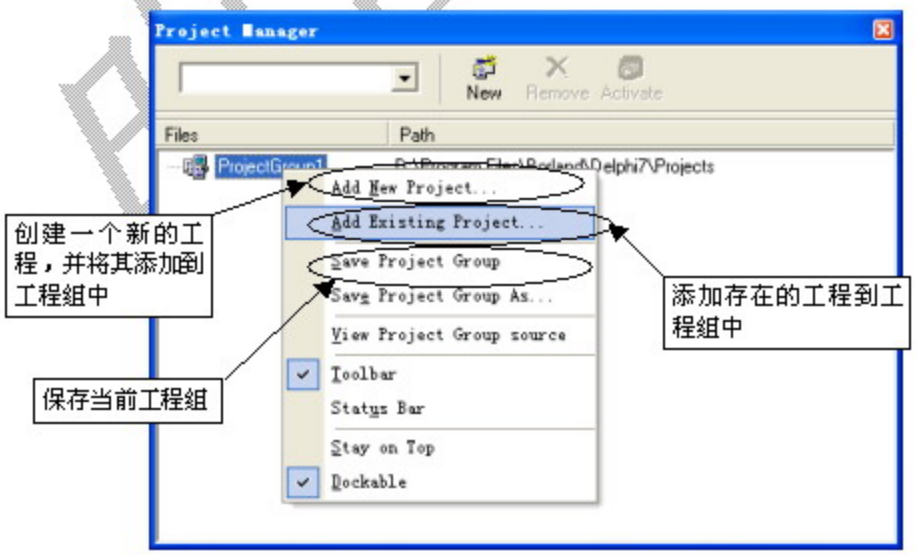


图 22.20 快捷菜单

（3）单击“Add Existing Project”选项，打开如图 22.21 所示的对话框。

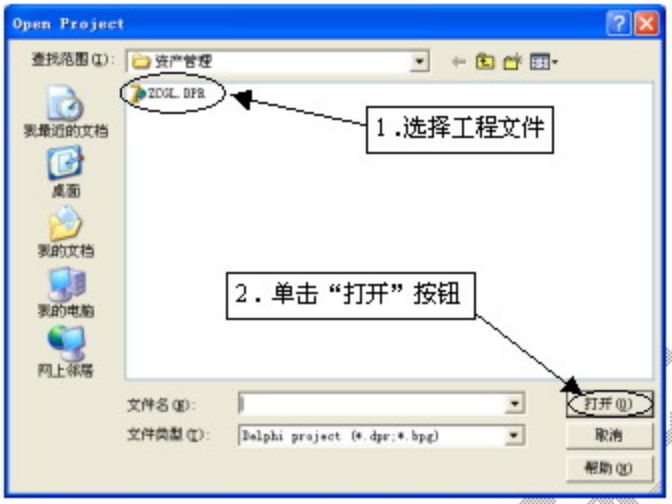


图 22.21　打开工程对话框

（4）选择工程文件，单击“打开”按钮，将项目文件添加到项目组中。

22.12.2　经验总结

在开发应用程序时，经常会遇到在一个单元（例如 Unit1）中需要访问另一个单元（例如 Unit2）中的内容，而在 Unit2 中也需要访问 Unit1 中的内容。此时，不能在 Unit1、Unit2 的“Uses”部分同时引用 Unit2 和 Unit1，因为这样做会造成循环引用的错误，但是，可以在其中一个单元的实现部分引用另一个单元。

22.13　系统的编译与发行

本系统编译生成的可执行文件为 1.7MB，如果要减少可执行文件的大小，可以使用压缩软件进行压缩。要在其他机器上运行本系统，需要具备以下几个条件。

- SQL Server 2000 数据库。
- 固定资产管理系统数据库。
- 本系统的可执行文件。

Delphi 在财务凭证管理系统中的应用

第 23 章

23.1 财务凭证管理系统概述

23.1.1 财务凭证管理系统简介

凭证是记录账簿的依据，它的完整与否将直接影响财务的核算结果。早期的财务凭证是由人工管理的，不仅容易出错，而且不易保存。随着计算机的普及和软件技术的不断发展，加上企业财务管理的迫切需要，凭证管理系统应运而生。

23.1.2 系统功能描述

- 实现摘要管理，操作员可以根据财务账单任意创建摘要。
- 完成凭证的录入、审核、过账。
- 实现凭证查询。

23.2 系 统 分 析

23.2.1 系统规划

根据上面的系统功能描述，可以定义如下需求分析。

1. 系统目标

使用计算机帮助财会人员完成凭证管理。

2. 系统处理范围

- 项目管理

- 凭证管理
- 凭证查询

3. 系统功能

- 摘要管理

在摘要管理中需要实现摘要的添加、修改、删除。

- 会计科目管理
- 凭证录入管理
- 凭证审核管理
- 凭证过账管理
- 查询管理

在查询管理中需要实现对历史凭证的查找。

4. 系统结构图

凭证管理系统的结构图如图 23.1 所示。

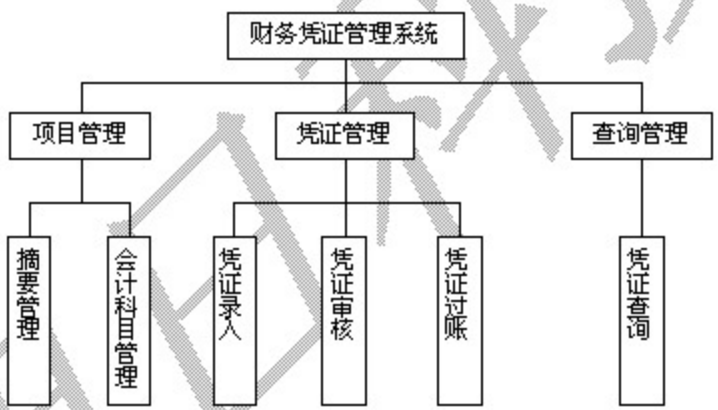


图 23.1 凭证管理系统结构图

23.2.2 业务流程

凭证管理系统的业务流程图如图 23.2 所示。

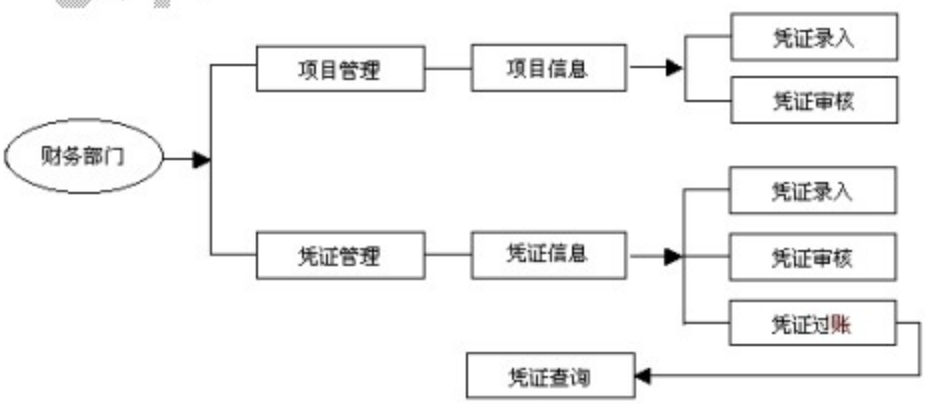


图 23.2 凭证管理系统业务流程图

23.3 数据库设计

23.3.1 数据库概念设计

凭证管理系统 E-R 图如图 23.3 所示。



图 23.3 凭证管理系统 E-R 图

23.3.2 数据库逻辑结构设计

凭证管理系统共使用了 5 张数据表，分别为：凭证表、凭证明细表、摘要表、会计科目表、凭证记账表。由于篇幅关系，在此只列出使用较多的数据表结构（如表 23.1 和表 23.2 所示），其他表结构请参考光盘。

表 23.1 凭证表

字段名称	字段类型	主键	外键	是否为空	默认值
凭证号	Varchar(30)	是		否	
借方金额	Money				
贷方金额	Money				
附单据	Int				
制单人	Varchar(30)				
审核人	Varchar(30)				
是否通过	Varchar(10)				否
是否过账	Varchar(10)				否
日期	DateTime				GetDate()

表 23.2 凭证明细表

字段名称	字段类型	主键	外键	是否为空	默认值
凭证号	Varchar(30)				
摘要编号	Varchar(30)				
摘要名称	Varchar(30)				
科目编号	Varchar(30)				
科目名称	Varchar(30)				
借方金额	Money				
贷方金额	Money				

23.4 处理过程分析

23.4.1 凭证录入处理过程分析

凭证录入是将凭证信息（凭证号、摘要、会计科目等）保存到数据表中，为了方便录入，凭证号应由程序自动产生，所以，首先需要进行凭证登记，产生凭证号，然后录入制单人、摘要、会计科目、借方金额、贷方金额等信息。在现代的会计电算化管理中，为了提高财会人员的录入速度，可以在摘要和会计科目中输入摘要编号和会计科目编号，由程序调出相应的摘要名称和会计科目名称。在本系统中，为了保证凭证的完整性，如果会计科目中的借方金额或贷方金额没有相应的贷方金额或借方金额，程序会提示项目列表有误，不允许保存。凭证录入业务流程如图 23.4 所示。

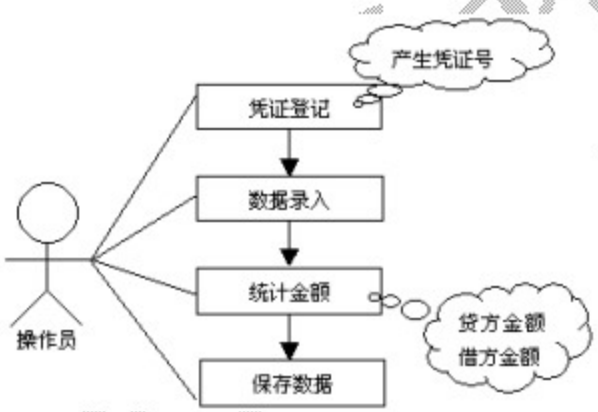


图 23.4 凭证录入业务流程

23.4.2 凭证审核处理过程分析

凭证审核是对凭证信息进行核实，确定是否可以过账。所以在审核前需要调出凭证信息。需要注意的是，这些凭证信息必须是没有审核或审核没有通过的凭证信息。为了使用户能够方便地调出凭证信息，在程序运行时，会将凭证号显示在一个下拉列表框中，用户可以在列表框中选择凭证号，也可以直接输入凭证号。凭证审核业务流程如图 23.5 所示。

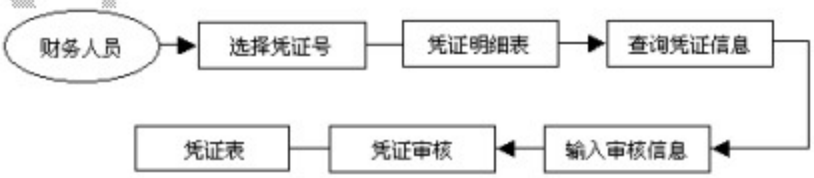


图 23.5 开发流程图

23.5 主窗体设计

主窗体设计步骤如下。

（1）新建一个窗体，在窗体上放置 TMainMenu、TImageList、TListView、TPanel、TImage 等组件，效果如图 23.6 所示。

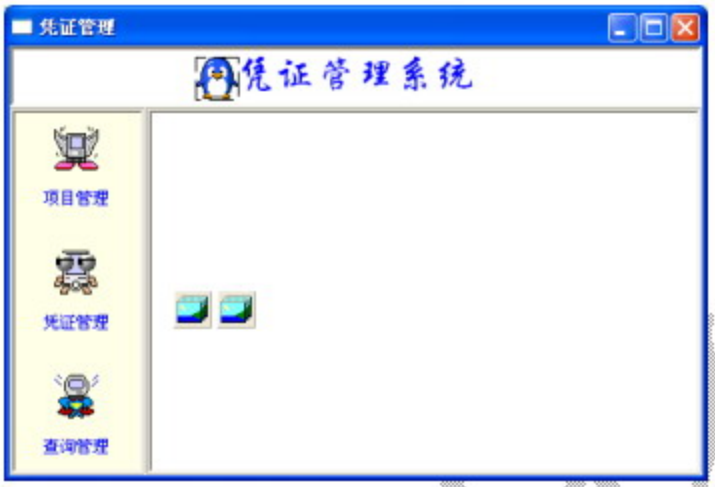


图 23.6　主窗体界面

（2）设置各主要组件属性如表 23.3 所示。

表 23.3　　组件属性设置

对　象　名	对 象 类 型	属　　性	值
Form1	TForm	Font.Color	clBlue
		Font.Size	8
ListView1	TListView	LargeImages	ImageList1

23.6　数据模块设计

数据模块设计步骤如下。

（1）启动 Dephi 7，打开工程“pz”，单击“File”/“New”/“Data Module”菜单项，创建数据模块。

（2）选中数据模块，在对象观察器中修改“Name”属性为“da”。

（3）在数据模块上放置 TADOConnection 组件、TADOQuery 组件，如图 23.7 所示。



图 23.7　数据模块

（4）设置各主要组件属性如表 23.4 所示。

表 23.4　　组件属性设置

对　象　名	对 象 类 型	属　　性	值
ADOConnection1	TADOConnection	ConnectionString	Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=凭证管理系统
ADOQuery1	TADOQuery	Connection	ADOConnection1
ADOQuery2	TADOQuery	Connection	ADOConnection1
ADOQuery3	TADOQuery	Connection	ADOConnection1

23.7 程序设计与编码

23.7.1 摘要管理程序设计

1. 概述

摘要是对凭证内容的简单概述。通过查看摘要，财务人员能够了解凭证的来源和作用。摘要管理为财务人员提供了设置摘要的功能，通过摘要管理，财务人员可以根据需要设置摘要内容。摘要管理运行结果如图 23.8 所示。



图 23.8 摘要管理运行窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TDataSource、TDBGrid、TEdit 等组件。

（2）设置各主要组件属性如表 23.5 所示。

表 23.5 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form3	TForm	Font.Color	ClBlue
		Font.Size	8
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

当窗体显示时，如果摘要表中有数据，会将摘要表中的数据显示出来，如果摘要表中没有数据，“删除”按钮将不可用。

```
procedure TForm3.FormShow(Sender: TObject);
begin
  with Da.ADOQuery1 do
  begin
    Close; //关闭数据集
    SQL.Clear; //清空SQL语句
    SQL.Add('select * from 摘要表'); //添加SQL语句
```

```
    Open; //打开数据集
  end;
  //将摘要表中内容显示在DBGrid1中
  if Da.ADOQuery1.RecordCount>0 then
    DataSource1.DataSet := Da.ADOQuery1
  else
  begin
    DataSource1.DataSet := Nil;
    BitBtn2.Enabled := False;
  end;
end;
```

单击“保存”按钮，会将摘要编号和摘要名称保存到数据库中，同时会将新添加的摘要信息显示在列表中，程序首先判断摘要编号和摘要名称是否为空，如果为空，不会进行插入操作。

```
procedure TForm3.BitBtn4Click(Sender: TObject);
begin
  if (Trim(Edit1.Text)<>'') and(Trim(Label3.Caption)<>'') then
  begin
    Try
      //向摘要表中插入数据
      with Da.ADOQuery2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('Insert  摘要表  values (:a,:b)');
        Parameters.ParamByName('a').Value := Trim(Label3.Caption);
        Parameters.ParamByName('b').Value := Trim(Edit1.Text);
        ExecSQL;
      end;
      Application.MessageBox('添加成功。','提示',64);
      BitBtn2.Enabled := True;
      BitBtn5.OnClick(Sender);
      Self.OnShow(Sender);
      BitBtn1.SetFocus;
    Except
      Application.MessageBox('系统出错。','提示',64);
      Close;
    end;
  end
  else
  begin
    Application.MessageBox('请添加摘要编号或摘要名称。','提示',64);
    BitBtn1.SetFocus;
  end;
end;
```

以下是“删除”按钮的单击事件，当用户单击“删除”按钮时，会出现提示对话框，

确认用户是否真的要删除当前数据。

```
procedure TForm3.BitBtn2Click(Sender: TObject);
begin
  if Da.ADOQuery1.RecordCount>0 then
    if Application.MessageBox(Pchar('确实要删除摘要编号为'+
          Trim(Da.ADOQuery1.FieldByName('摘要编号').Value)+'的摘要吗？'),'提示',MB_YesNo)=
          ID_Yes then
    begin
      Da.ADOQuery1.Delete;
      self.OnShow(Sender);
    end;
end;
```

23.7.2 凭证录入程序设计

1. 概述

凭证录入是进行凭证管理的起点，而且也是最繁琐的工作。操作员在录入凭证时，一定要保证财务账目的准确性和凭证的完整性，同时程序也会对凭证的完整性做相应检查，例如，如果会计科目只有贷方金额没有借方金额，或只有借方金额没有贷方金额，在保存时，会提示项目列表有误。如果操作员需要在贷方金额或借方金额处输入负数，可以在贷方金额或借方金额最前面的单元格中输入“-”。凭证录入运行结果如图 23.9 所示。

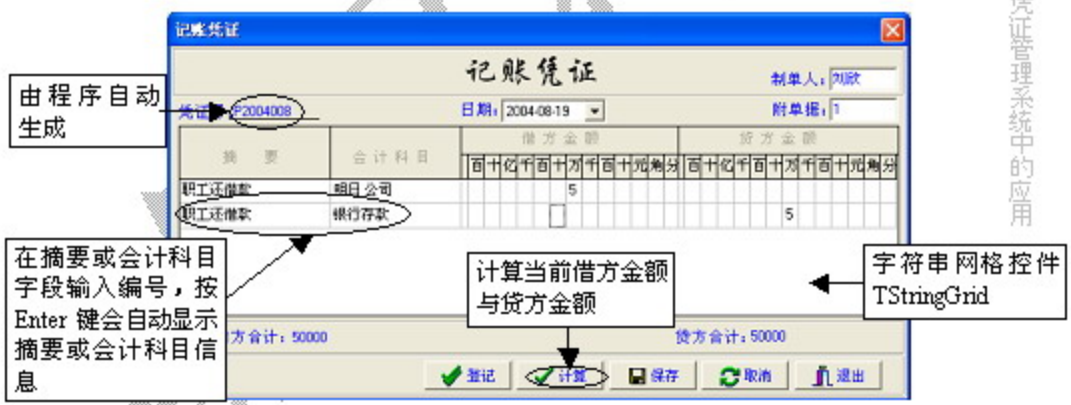


图 23.9 凭证录入界面

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TLabel、TEdit、TDataTimePicker、TMemo 等组件。

（2）设置各主要组件属性如表 23.6 所示。

表 23.6 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form2	TForm	Font.Color	clBlue
		Font.Size	8
BitBtn4	TBitBtn	Kind	bkClose

3. 程序相关代码

在窗体显示时设置 TStringGrid 列标题，同时在 DateTimePicker1 中显示系统当前时间。

```
procedure TForm2.FormShow(Sender: TObject);
begin
  //设置列标题
  a.Cells[3,0]:= '百';
  a.Cells[4,0]:='十';
  a.Cells[5,0]:='亿';
  a.Cells[6,0]:='千';
  a.Cells[7,0]:='百';
  a.Cells[8,0]:='十';
  a.Cells[9,0]:='万';
  a.Cells[10,0]:='千';
  a.Cells[11,0]:='百';
  a.Cells[12,0]:='十';
  a.Cells[13,0]:='元';
  a.Cells[14,0]:='角';
  a.Cells[15,0]:='分';
  a.Cells[17,0]:= '百';
  a.Cells[18,0]:='十';
  a.Cells[19,0]:='亿';
  a.Cells[20,0]:='千';
  a.Cells[21,0]:='百';
  a.Cells[22,0]:='十';
  a.Cells[23,0]:='万';
  a.Cells[24,0]:='千';
  a.Cells[25,0]:='百';
  a.Cells[26,0]:='十';
  a.Cells[27,0]:='元';
  a.Cells[28,0]:='角';
  a.Cells[29,0]:='分';
  DateTimePicker1.Date := Now();
end;
```

下面代码主要功能是防止在贷方金额和借方金额中输入非法字符，只能输入整数。此外，当一个摘要有借方金额，则该行不能输入贷方金额，同理，当一个摘要有贷方金额，则该行不能再输入借方金额。

```
procedure TForm2.aKeyPress(Sender: TObject; var Key: Char);
var
  aa: Boolean;
  i: Integer;
begin
  if y>15 then //当一个摘要有借方金额，则该行不能输入贷方金额
    For i :=2 to 15 do
    begin
      if Trim(a.Cells[i,x])<>'' then
```

```
            Key := #0;
        end;
    if (y>1)and(y<16) then//当一个摘要有贷方金额，则该行不能再输入借方金额
        For i :=17 to 29 do
        begin
            if Trim(a.Cells[i,x])<>'' then
                Key := #0;
        end;
    if (y = 2)or(y= 16) then
        if Key <> #45 then
            Key := #0;
    if (y>2)and(y<>16)then
    begin
        aa := (Key<#8)or(Key>#8)and(Key<#48)or(Key>#57);
        if aa then
            Key := #0;
        if (a.Cells[y,x]<>'')and(Key<>#8) then
            Key := #0;
    end;
end;
```

下面是字符串网格的 OnKeyDown 事件代码，操作员在字符串网格中按 Delete 键会删除当前行数据，按 PageDown 键会在字符串网格中添加新行，按 Enter 键会将操作员录入的摘要编号或会计科目编号转换成摘要名称或会计科目名称，前提是摘要编号或会计科目编号必须正确。

```
procedure TForm2.aKeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
var
    mm,nn: Integer;
begin
    if Key = VK_Delete then //按Delete键可以删除当前行
        if Application.MessageBox('确实要删除当前行数据吗？','提示',MB_YesNo)= ID_Yes then
        begin
            if a.RowCount>2 then
            begin
                ClearCurrent; //清空当前行数据
                if x<>a.RowCount-1 then
                begin
                    For mm := x +1 to a.RowCount-1 do //利用双重循环将当前行之后的数据赋给上一行数据
                        For nn := 0 to a.ColCount-1 do
                            a.Cells[nn,mm-1] := a.Cells[nn,mm];
                end;
                ClearEnd;//清空最后一行数据
                a.RowCount := a.RowCount-1;//使表格行数减一
            end
            else if a.RowCount = 2 then
                ClearCurrent;
```

```
        Label8.Caption := '';
        Label10.Caption := '';
      end;
  if Key = VK_Next then //按PageDown键添加新行
    if (x = a.RowCount-1)and(Trim(a.Cells[0,x])<>'')and(Trim(a.Cells[1,x])<>'')
      and((JIsNull = False)or(DIsNull = False)) then
    begin
      a.RowCount := a.RowCount +1;
      a.Cells[0,x+1]:= a.Cells[0,x];
    end;
  if Key = VK_ReTurn then
  begin
    if (y = 0)and(Trim(a.Cells[y,x])<>'') then
    begin
      with Da.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 摘要表 where 摘要编号 = :a or 摘要名称 = :b');
        Parameters.ParamByName('a').Value := Trim(a.Cells[y,x]);
        Parameters.ParamByName('b').Value := Trim(a.Cells[y,x]);
        Open;
        if Da.ADOQuery1.RecordCount>0 then
          a.Cells[y,x]:= Da.ADOQuery1.FieldByName('摘要名称').Value
        else
        begin
          Application.MessageBox('该摘要不存在，请重新输入。','提示',64);
          a.Cells[y,x]:='';
        end;
      end;
    end
    else if (y = 1)and(Trim(a.Cells[y,x])<>'') then
    begin
      With Da.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 会计科目表 where 科目编号 = :a or 科目名称 = :b');
        Parameters.ParamByName('a').Value := Trim(a.Cells[y,x]);
        Parameters.ParamByName('b').Value := Trim(a.Cells[y,x]);
        Open;
        if Da.ADOQuery1.RecordCount>0 then
          a.Cells[y,x]:= Da.ADOQuery1.FieldByName('科目名称').Value
        else
        begin
          Application.MessageBox('该摘要不存在，请重新输入。','提示',64);
          a.Cells[y,x]:='';
```

```
            end;
          end;
        end
    end;
end;
```

函数 JisNull 判断当前行借方金额是否为空，如果为空，函数返回 True，否则返回 False。

```
Function TForm2.JIsNull: Boolean;
var
  i: Integer;
begin
  JIsNull := True;
  For i := 2 to 14 do
  begin
    if Trim(a.Cells[i,x])<>'' then
    begin
      JIsNull := False;
      Break;
    end;
  end;
end;
```

如果对于每个借方数据都有对应的贷方数据，并且对于每一个贷方数据都有其对应的借方数据则 DPJD 为 True，否则为 False。

```
function TForm2.DPJD: Boolean;
var
  m,n: Integer;
  mm,nn: Boolean;
begin
  DPJD := False;
  mm := True; //如果对于每个借方数据都有对应的贷方数据则mm = True
  nn := True; //如果对于每个贷方数据都有对应的借方数据则nn = True
  for m := 1 to a.RowCount-1 do
    if (JIsNull(m)= False)and(DIsCz(Trim(a.Cells[0,m]))= False) then
    begin
      mm := False;
      Break;
    end;
  For n := 1 to a.RowCount-1 do
    if (DisNull(n)= False)and(JIsCz(Trim(a.Cells[0,n]))= False)then
    begin
      nn := False;
      Break;
    end;
  if (mm = True)and(nn = True) then
    DPJD := True;
end;
```

函数 JisNull 判断指定行借方金额是否为空，如果为空，函数返回 True，否则返回 False。

```
function TForm2.JIsNull(Row: Integer): Boolean;
var
  i: Integer;
begin
  JIsNull := True;
  For i := 3 to 15 do
    if Trim(a.Cells[i,Row])<>'' then
    begin
      JIsNull := False;
      Break;
    end;
end;
```

下面是“保存”按钮的单击事件代码，单击“保存”按钮，程序会检查各个数据项是否为空，如果不为空，会将凭证信息保存到凭证表和凭证明细表中。

```
procedure TForm2.BitBtn2Click(Sender: TObject);
var
  mm: Integer;
begin
  if Trim(Label3.Caption)='' then
  begin
    Application.MessageBox('凭证号不能为空。','提示',64);
    BitBtn1.SetFocus;
    Exit;
  end;
  if Trim(Edit3.Text)='' then
  begin
    Application.MessageBox('制单人不能为空。','提示',64);
    Edit3.SetFocus;
    Exit;
  end;
  if Trim(Edit4.Text)='' then
  begin
    Application.MessageBox('制单人不能为空。','提示',64);
    Edit4.SetFocus;
    Exit;
  end;
  if StrToInt(Edit4.Text)<1 then
  begin
    Application.MessageBox('附单拒不能小于1。','提示',64);
    Edit4.SetFocus;
    Exit;
  end;
  if DPJD = False then
  begin
    Application.MessageBox('项目列表有误。','提示',64);
```

```
    Exit;
  end;
  if (Trim(Label8.Caption)='')or(Trim(Label10.Caption)='') then
  begin
    Application.MessageBox('请计算贷方金额或借方金额。','提示',64);
    BitBtn5.SetFocus;
    Exit;
  end;
  if JCZYKJ = False then
  begin
    Application.MessageBox('列表中摘要或会计不存在。','提示',64);
    Exit;
  end;
  Try
    Da.ADOConnection1.BeginTrans;
    with da.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('insert 凭证表 values(:a,:b,:c,:d,:e,default,Default,Default,:f)');
      Parameters.ParamByName('a').Value := Trim(Label3.Caption);
      Parameters.ParamByName('b').Value := StrToFloat(Label8.Caption);
      Parameters.ParamByName('c').Value := StrToFloat(Label10.Caption);
      Parameters.ParamByName('d').Value := StrToInt(Edit4.Text);
      Parameters.ParamByName('e').Value := Trim(Edit3.Text);
      Parameters.ParamByName('f').Value := DateTimePicker1.Date;
      ExecSQL;
    end;
    For mm := 1 to a.RowCount-1 do
    begin
      with Da.ADOQuery1 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('insert 凭证明细表 values (:a,:b,:c,:d,:e,:f,:g)');
        Parameters.ParamByName('a').Value := Trim(Label3.Caption);
        with Da.ADOQuery2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('select * from 摘要表 where 摘要名称 = :a');
          Parameters.ParamByName('a').Value := a.Cells[0,mm];
          Open;
        end;
        Parameters.ParamByName('b').Value := Trim(Da.ADOQuery2.FieldByName('摘要编号
        ').Value);
        Parameters.ParamByName('c').Value := Trim(a.Cells[0,mm]);
```

```
            with Da.ADOQuery2 do
            begin
              Close;
              SQL.Clear;
              SQL.Add('select * from 会计科目表 where 科目名称 = :a');
              Parameters.ParamByName('a').Value := Trim(a.Cells[1,mm]);
              Open;
            end;
            Parameters.ParamByName('d').Value := Trim(Da.ADOQuery2.FieldByName('科目编号
            ').Value);
            Parameters.ParamByName('e').Value := Trim(a.Cells[1,mm]);
            Parameters.ParamByName('f').Value := jsJFje(mm);
            Parameters.ParamByName('g').Value := JSDFje(mm);
            ExecSQL;
          end;
        end;
        Da.ADOConnection1.CommitTrans;
        Application.MessageBox('操作成功。','提示',64);
        BitBtn3.OnClick(Sender);
      Except
        Da.ADOConnection1.RollbackTrans;
        Application.MessageBox('操作失败。','提示',64);
      end;
    end;
```

23.7.3 凭证审核程序设计

1. 概述

凭证审核是会计业务中的一项重要工作，审核人员按照有关会计法规、制度对已经输入系统的凭证进行检查、核对，以保证财务账目的准确性。在凭证管理系统中，凭证审核主要有两个目的，一是发现录入过程中无法发现的错误，二是为记账凭证做一个标记，标识记账凭证是否通过审核，如果凭证审核没有通过，将不允许过账。在凭证审核前，需要查询预审核的凭证信息。凭证审核运行结果如图 23.10 所示。



图 23.10 凭证审核窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TSpeedButton、TLabel、TEdit、TComboBox、TDataSource、TDBGrid、TCheckBox 等组件。

（2）设置各主要组件属性如表 23.7 所示。

表 23.7　　组件属性设置

对 象 名	对 象 类 型	属　性	值
Form5	TForm	Font.Color	clBlue
		Font.Size	10
DBGrid1	TDBGrid	DataSource	DataSource1
CheckBox1	TCheckBox	Checked	False

3. 程序相关代码

下面是“取消”按钮的单击事件代码，主要完成界面和数据的初始化工作。

```
procedure TForm5.BitBtn2Click(Sender: TObject);
begin
  CheckBox1.Checked := False; //将CheckBox1置为非选中状态
  ComboBox2.Text := '';
  DataSource1.DataSet := Nil;
  BitBtn1.Enabled := False; //将“审核”按钮设置为不可用
  ComboBox1.Clear; //清空ComboBox1中的凭证号
  with Da.ADOQuery1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('select 凭证号 from 凭证表 where 是否通过 = "否"');
    Open;
  end;
  //向ComboBox1中添加凭证号
  if Da.ADOQuery1.RecordCount>0 then
    while Not Da.ADOQuery1.Eof do
    begin
      ComboBox1.Items.Add(Da.ADOQuery1.FieldByName('凭证号').Value);
      Da.ADOQuery1.Next;
    end;
end;
```

单击“查询”按钮会查询操作员输入或选择的凭证信息，程序首先判断是否进行批量查询，如果进行批量查询，会显示凭证明细表中的相关凭证信息，如果不进行批量查询，会按照操作员输入或选择的凭证号进行查询。

```
procedure TForm5.SpeedButton1Click(Sender: TObject);
begin
  if CheckBox1.Checked = False then   //判断是否进行批量查询
  begin
    with Da.ADOQuery2 do
```

```
        begin
          Close;
          SQL.Clear;
          SQL.Add('select * from 凭证明细表 where 凭证号 = :a');
          Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
          Open;
        end;
        if Da.ADOQuery2.RecordCount>0 then
        begin
          DataSource1.DataSet := da.ADOQuery2;
          BitBtn1.Enabled := True;
        end
        else
        begin
          BitBtn1.Enabled := False;
          Application.MessageBox('该凭证数据不存在。','提示',64);
          DataSource1.DataSet := Nil;
        end;
      end
      else
      begin
        with Da.ADOQuery2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('select * from 凭证明细表 where 凭证号 in ( Select 凭证号 from 凭证表 where
          是否通过 = "否")');
          Open;
        end;
        if Da.ADOQuery2.RecordCount>0 then
        begin
          DataSource1.DataSet := da.ADOQuery2;
          BitBtn1.Enabled := True;
        end
        else
        begin
          BitBtn1.Enabled := False;
          Application.MessageBox('该凭证数据不存在。','提示',64);
          DataSource1.DataSet := Nil;
        end;
      end;
    end;
```

以下是“审核”按钮的单击事件代码，程序首先判断操作员是否输入了审核人、审核是否通过，如果没有填写，不进行任何操作，然后程序根据操作员在是否通过下拉列表中选择的不同选项进行不同的处理。如果凭证通过审核，将凭证表中的“是否通过”字段设置为“是”，以标识该凭证已经通过审核。

```
procedure TForm5.BitBtn1Click(Sender: TObject);
begin
  if (Trim(Edit1.Text)<>'')and(Trim(ComboBox2.Text)<>'') then
    if Trim(ComboBox2.Text)='是' then
    begin
      Try //异常捕捉
      Da.ADOConnection1.BeginTrans; //开始一个事务
      Da.ADOQuery2.First;//将记录集当前行定位到第一行数据上
      while Not Da.ADOQuery2.Eof do //利用循环更改凭证表中的数据
      begin
        with da.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('update 凭证表 set 审核人 = :a , 是否通过 = "是" where 凭证号 = :b');
          Parameters.ParamByName('a').Value := Trim(Edit1.Text);
          Parameters.ParamByName('b').Value := Trim(Da.ADOQuery2.FieldByName('凭证号
          ').Value);
          ExecSQL;
        end;
        Da.ADOQuery2.Next;
      end;
        Da.ADOConnection1.CommitTrans;//开始一个事务
        Application.MessageBox('操作成功,审核通过。','提示',64);
        BitBtn1.Enabled := False;
        BitBtn2.OnClick(Sender);
      Except
        Da.ADOConnection1.RollbackTrans;//回滚事务
        Application.MessageBox('操作失败。','提示',64);
      end;
    end
    else if Trim(ComboBox2.Text)='否' then
    begin
      while Not Da.ADOQuery2.Eof do
      begin
        with da.ADOQuery1 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('update 凭证表 set 审核人 = :a  where 凭证号 = :b');
          Parameters.ParamByName('a').Value := Trim(Edit1.Text);
          Parameters.ParamByName('b').Value := Trim(Da.ADOQuery2.FieldByName('凭证号
          ').Value);
          ExecSQL;
        end;
        Da.ADOQuery2.Next;
      end;
```

```
        Application.MessageBox('操作成功，审核没有通过。','提示',64);
        BitBtn1.Enabled := False;
        BitBtn2.OnClick(Sender);
      end;
end;
```

23.7.4 凭证过账程序设计

1. 概述

凭证过账是进行凭证处理的最后一个过程。它是对审核通过的记账凭证进行登记入账。经过登记处理后的凭证将成为历史凭证，不允许修改。在凭证过账时，需要查询凭证信息，凭证信息必须是审核通过的没有过账的凭证。凭证过账运行结果如图 23.11 所示。

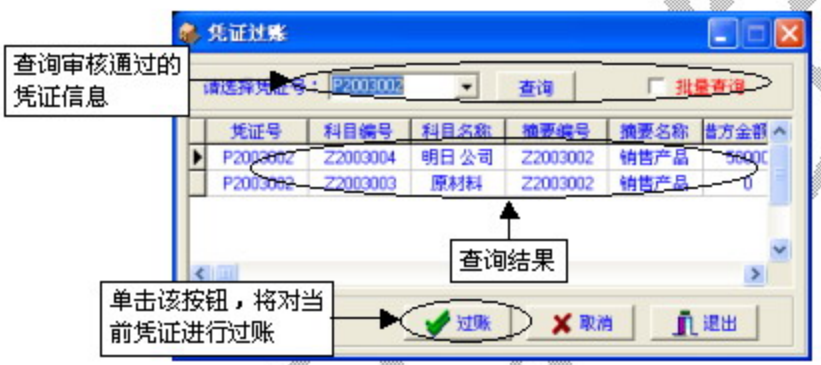


图 23.11 凭证过账窗口

2. 设计步骤

（1）新建一个窗体，在窗体上放置 TPanel、TBitBtn、TSpeedButton、TLabel、TCheckBox、TDataSource、TDBGrid 等组件。

（2）设置各主要组件属性如表 23.8 所示。

表 23.8 组件属性设置

对 象 名	对 象 类 型	属 性	值
Form6	TForm	Font.Color	clBlue
		Font.Size	8
CheckBox1	TCheckBox	Checked	False
DBGrid1	TDBGrid	DataSource	DataSource1

3. 程序相关代码

下面是“查询”按钮的单击事件代码，单击“查询”按钮会返回凭证明细表中审核通过并没有过账的凭证信息。程序首先确定是否进行批量查询，即 CheckBox1 是否被选中，如果进行批量查询，则查询所有通过审核而没有进行过账的凭证信息，如果不进行批量查询，程序根据用户输入的凭证号查询。

```
procedure TForm6.SpeedButton1Click(Sender: TObject);
begin
```

```
    if CheckBox1.Checked = False then
    begin
      with Da.ADOQuery2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 凭证明细表 where 凭证号 = :a');
        Parameters.ParamByName('a').Value := Trim(ComboBox1.Text);
        Open;
      end;
      if Da.ADOQuery2.RecordCount>0 then
      begin
        DataSource1.DataSet := da.ADOQuery2;
        BitBtn1.Enabled := True;
      end
      else
      begin
        BitBtn1.Enabled := False;
        Application.MessageBox('该凭证数据不存在。','提示',64);
        DataSource1.DataSet := Nil;
      end;
    end
    else
    begin
      with Da.ADOQuery2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add('select * from 凭证明细表 where 凭证号 in( Select 凭证号 from 凭证表 where
        是否过账 = "否" and 是否通过 = "是")');
        Open;
      end;
      if Da.ADOQuery2.RecordCount>0 then
      begin
        DataSource1.DataSet := da.ADOQuery2;
        BitBtn1.Enabled := True;
      end
      else
      begin
        BitBtn1.Enabled := False;
        Application.MessageBox('该凭证数据不存在。','提示',64);
        DataSource1.DataSet := Nil;
      end;
    end;
  end;
```

以下是“过账”按钮的单击事件代码。在进行凭证过账时，主要将凭证表中的“是否过账”标识设置为“是”，然后向凭证记账表中保存过账信息。由于在过账过程中既要修改

凭证表信息，又要向凭证记账表中插入数据，为保证该操作的完整性，在程序中使用了事务进行控制。

```
procedure TForm6.BitBtn1Click(Sender: TObject);
begin
  Try
  Da.ADOConnection1.BeginTrans; //开始一个事务
  Da.ADOQuery2.First;
  While Not Da.ADOQuery2.Eof do
  begin
    with Da.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('update 凭证表 set 是否过账 = "是" where 凭证号 = :a');
      Parameters.ParamByName('a').Value := Trim(Da.ADOQuery2.FieldByName('凭证号
      ').Value);
      ExecSQL;
    end;
    with Da.ADOQuery1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('insert 凭证记账表 Values (:a,:b,:c,:d,:e,:f,:g)');
      Parameters.ParamByName('a').Value := Trim(Da.ADOQuery2.FieldByName('凭证号
      ').Value);
      Parameters.ParamByName('b').Value := Trim(Da.ADOQuery2.FieldByName('科目编号
      ').Value);
      Parameters.ParamByName('c').Value := Trim(Da.ADOQuery2.FieldByName('科目名称
      ').Value);
      Parameters.ParamByName('d').Value := Trim(Da.ADOQuery2.FieldByName('摘要编号
      ').Value);
      Parameters.ParamByName('e').Value := Trim(Da.ADOQuery2.FieldByName('摘要名称
      ').Value);
      Parameters.ParamByName('f').Value := Da.ADOQuery2.FieldByName('借方金额').Value;
      Parameters.ParamByName('g').Value := Da.ADOQuery2.FieldByName('贷方金额').Value;
      ExecSQL;
    end;
    Da.ADOQuery2.Next;
  end;
    Da.ADOConnection1.CommitTrans; //提交事务
    Application.MessageBox('操作成功。','提示',64);
    BitBtn2.OnClick(Sender);
  Except
    Da.ADOConnection1.RollbackTrans; //回滚事务
    Application.MessageBox('操作失败。','提示',64);
  end;
end;
```

23.8 疑难问题分析解决

23.8.1 如何实现函数重载

函数重载允许几个函数使用相同的函数名。在程序执行时，会根据参数个数或参数类型来判断具体调用哪个函数。请看下面的代码：

```
Function JIsNull: Boolean;overload; //函数声明
Function JIsNull(Row: Integer): Boolean;overload;//函数声明
Function TForm2.JIsNull: Boolean;
var
  i: Integer;
begin
  JIsNull := True;
  For i := 2 to 14 do
  begin
    if Trim(a.Cells[i,x])<>'' then
    begin
      JIsNull := False;
      Break;
    end;
  end;
end;
function TForm2.JIsNull(Row: Integer): Boolean;
var
  i: Integer;
begin
  JIsNull := True;
  For i := 3 to 15 do
    if Trim(a.Cells[i,Row])<>'' then
    begin
      JIsNull := False;
      Break;
    end;
end;
```

当在程序中使用“JisNull(10)”语句调用函数时，程序会调用第二个函数，如果采用“JisNull”语句调用函数，程序会调用第一个函数。

23.8.2 如何解决两个整数乘积超出整数范围

在开发应用程序时，可能会遇到这种情况，当两个整数的乘积超出了整数范围时，其结果并不是我们所期望的。例如：

```
StrToInt(a.Cells[8,1])*1000000000;
```

执行结果可能会不正确，可以将 StrToInt()函数换成 StrToInt64()或者将一个整数转换成

实数。例如：

```
StrToInt(a.Cells[8,i])*1000000000.0;
StrToInt64(a.Cells[8,i])*1000000000;
```

23.8.3 如何向列表视图中添加选项

在本章程序的主窗体模块，当用户在左边的列表视图中选中一个项目时，在右边的列表视图中会显示相应的选项。其实，当用户在左边的列表视图中选中一个项目时，程序会清空右边的列表视图中的项目，然后添加相应的项目。向列表视图中添加选项可以有两种方法，一是在设计时通过对象观察器向列表视图中添加项目，二是通过程序代码向列表视图中添加项目。

- 通过对象观察器向列表视图中添加项目。具体操作如图 23.12 所示。

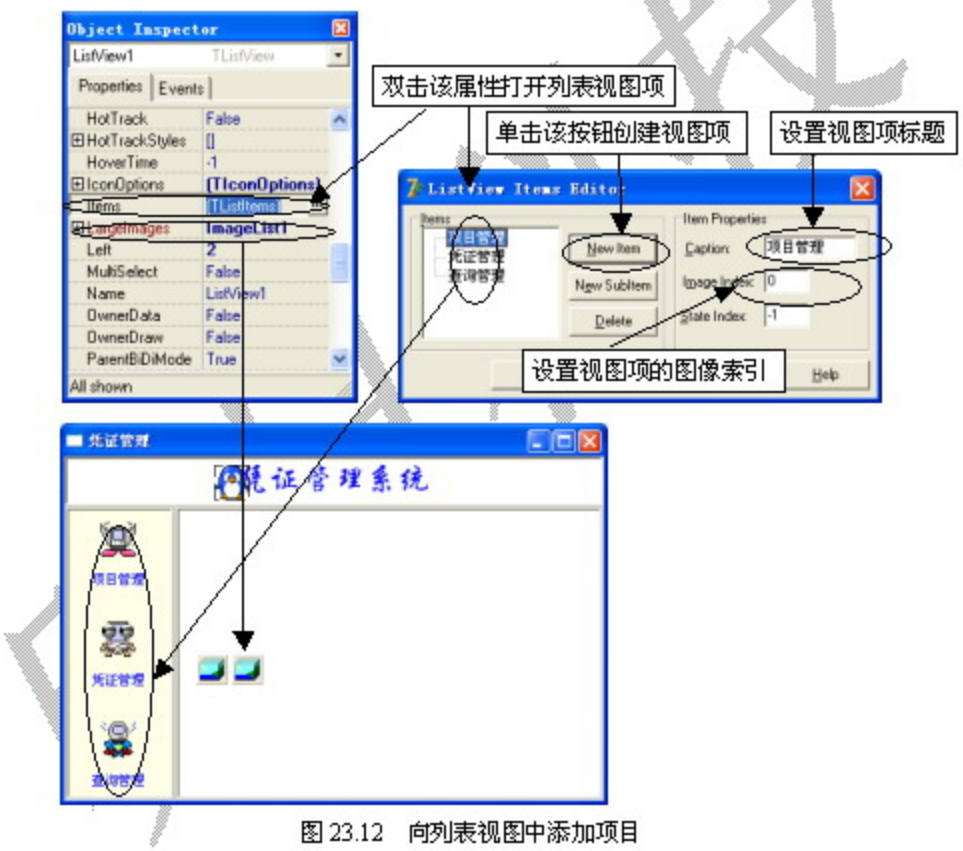


图 23.12 向列表视图中添加项目

- 通过程序代码向列表视图中添加项目

使用代码向列表视图中添加项目与使用对象观察器步骤相同，首先添加视图项，然后设置视图项标题，最后设置视图项的图像索引。代码如下：

```
//向列表视图中添加选项
procedure TForm1.AddItem(Caption: String;ImageIndex: Integer);
var
  ListItem1: TListItem;
begin
```

```
    ListItem1 := ListView3.Items.Add;
    ListItem1.Caption := Caption;
    ListItem1.ImageIndex := ImageIndex;
end;
```

23.8.4 如何响应窗体的 OnKeyDown 事件

当窗体被其他控件覆盖后，不通过 Tab 键，窗体很难获得焦点。因此，很多时候无法触发窗体的 OnKeyDown 事件。如果用户想要实现按一键（例如 Esc 键）退出程序，需要处理每一个控件的 OnKeyDown 事件。其实，在窗体中有一个“KeyPreview”属性，只要将该属性设为“True”，无论当前焦点处在哪个控件上，用户按键时，都会触发窗体的 OnKeyDown 事件。代码如下：

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if key = vk_ESCAPE then
    Close;
end;
```

23.8.5 凭证录入的正确性检查

凭证录入是最常用的模块之一，也是整个凭证管理系统的入口。凭证的正确与否，将直接影响到账表。所以，在设计时需要有健全的检查措施。

- 凭证号的检查

凭证号是记账凭证的标记，反映了经营业务发生的顺序，不能够重复。

- 会计科目的合法性检查

用户在输入会计科目时，科目代码或科目名称是否在会计科目表中存在。

- 金额检查

任何一张凭证必须满足“有借必有贷”的原则，既有借方金额，又有贷方金额，并且，借、贷合计金额必须相等。

23.9 系统测试

系统测试是程序开发过程中的最后一个环节，它是对程序员开发出来的软件进行整体测试，检查软件是否实现了客户的需求，软件是否存在 Bug。实际上，一个软件不存在 Bug 几乎是不可能的，只是程序开发人员、测试人员或者用户没有发现，或者在软件设计时，Bug 已经存在，而开发人员没有发现，以至于到软件完成后无法去除 Bug。下面将以测试本系统的凭证录入管理为例介绍测试程序的一般方法。

1. 按照操作流程测试

（1）运行程序，打开凭证录入窗口。效果如图 23.13 所示。

（2）单击“登记”按钮，查看程序是否自动生成了凭证号。如果没有生成凭证号，表

明程序没有实现预定的功能，这应该是一个比较严重的错误，测试人员应该记录高级别错误，但通常情况下，这类错误在测试人员测试时很少发生，因为程序开发人员在完成该功能实现代码后，都会进行这项测试，如果程序生成了凭证号，但有错误对话框弹出，此时，测试人员也应给予比较高的级别错误，然后将信息反馈给程序开发人员，程序开发人员根据错误信息重新修改程序。



图 23.13 凭证录入

（3）输入制单人，在摘要中输入摘要编号，例如“Z2003001”，然后按 Enter 键，查看程序是否显示摘要名称，同样在会计科目中也输入会计科目编号，例如“Z2003001”，按 Enter 键，查看程序是否显示会计科目名称。

（4）在借方金额中输入数据，然后在贷方金额中输入数据，查看是否可以在贷方金额中输入数据，因为在会计凭证中，一行数据不能同时存在借方金额和贷方金额。

（5）按 PageDown 键，看在表格中是否可以添加新行。按 Delete 键，看在表格中是否删除当前行。

（6）单击“计算”按钮，查看程序是否统计了借方金额和贷方金额，统计数据是否正确。

（7）单击“保存”按钮，查看程序是否将数据保存到数据表中。

（8）重新录入数据，单击“取消”按钮，查看是否清空了操作员所录入的数据。

2. 按照非操作流程测试

（1）运行程序，打开凭证录入窗口。效果如图 23.13 所示。

（2）随意单击“计算”、“保存”、“取消”按钮，查看程序是否有错误提示。

（3）在摘要和会计科目中随意输入数据，然后按 Enter 键，检查程序是否有错误。

（4）在贷方金额和借方金额中输入字符，例如“a”，检查程序是否允许输入。

（5）在凭证窗口中输入多行数据，然后随意删除数据，单击“计算”按钮，查看计算结果是否正确。

23.10 程序设计清单

凭证管理系统文件清单如表 23.9 所示。

表 23.9　　凭证管理系统设计清单

文 件 名	文 件 类 型	说　明
DataModal4.dfm	窗体文件	数据模块
Kmgl.dfm	窗体文件	会计科目管理窗体
Pzcx.dfm	窗体文件	凭证查询窗体
Pzgl.dfm	窗体文件	主窗体
Pzgz.dfm	窗体文件	凭证过账窗体
Pzlr.dfm	窗体文件	凭证录入窗体
Pzsh.dfm	窗体文件	凭证审核窗体
Zygl.dfm	窗体文件	摘要管理窗体
Pz.dpr	工程文件	
Pz.exe	可执行文件	

23.11　技术、经验总结

23.11.1　技术总结

在本系统的凭证录入模块中，用户单击“计算”按钮，程序会将 TStringGrid 中的贷方金额和借方金额数据统计出来，请看下面的程序代码：

```
procedure TForm2.BitBtn5Click(Sender: TObject);
var
  i: integer;
begin
  Label8.Caption := '';
  Label10.Caption := '';
  //初始化全局变量a1,a2
 a1 := 0.0;
  a2 := 0.0;
  //统计借方金额
  For i := 1 to a.RowCount-1 do
  Begin
    //如果用户在借方金额或贷方金额前输入“-”，统计结果将是负数
    if a.Cells[2,i]='-' then
    begin
      if a.Cells[3,i]<>'' then a1 := a1 -StrToInt(a.Cells[3,i])*10000000000.0;
      if a.Cells[4,i]<>'' then a1 :=a1 - StrToInt(a.Cells[4,i])*1000000000.0;
      if a.Cells[5,i]<>'' then a1 := a1 - StrToInt(a.Cells[5,i])*100000000.0;
      if a.Cells[6,i]<>'' then a1 := a1 - StrToInt(a.Cells[6,i])*10000000.0;
      if a.Cells[7,i]<>'' then a1 := a1 - StrToInt(a.Cells[7,i])*1000000.0;
      if a.Cells[8,i]<>'' then a1 := a1 - StrToInt(a.Cells[8,i])*100000.0;
      if a.Cells[9,i]<>'' then a1 := a1 - StrToInt(a.Cells[9,i])*10000.0;
      if a.Cells[10,i]<>'' then a1 := a1 - StrToInt(a.Cells[10,i])*1000.0;
      if a.Cells[11,i]<>'' then a1 := a1 - StrToInt(a.Cells[11,i])*100.0;
```

```
            if a.Cells[12,i]<>'' then a1 := a1 - StrToInt(a.Cells[12,i])*10.0;
            if a.Cells[13,i]<>'' then a1 := a1 - StrToInt(a.Cells[13,i]);
            if a.Cells[14,i]<>'' then a1 := a1 - StrToInt(a.Cells[14,i])*0.1;
            if a.Cells[15,i]<>'' then a1 := a1 -StrToInt(a.Cells[15,i])*0.01;
        end
        else
        begin
            if a.Cells[3,i]<>'' then a1 := a1 +StrToInt(a.Cells[3,i])*10000000000.0   ;
            if a.Cells[4,i]<>'' then a1 :=a1 + StrToInt(a.Cells[4,i])*1000000000.0;
            if a.Cells[5,i]<>'' then a1 := a1 +StrToInt(a.Cells[5,i])*100000000.0 ;
            if a.Cells[6,i]<>'' then a1 := a1 +StrToInt(a.Cells[6,i])*10000000.0;
            if a.Cells[7,i]<>'' then a1 := a1 +StrToInt(a.Cells[7,i])*1000000.0;
            if a.Cells[8,i]<>'' then a1 := a1 +StrToInt(a.Cells[8,i])*100000.0;
            if a.Cells[9,i]<>'' then a1 := a1 +StrToInt(a.Cells[9,i])*10000.0;
            if a.Cells[10,i]<>'' then a1 := a1 + StrToInt(a.Cells[10,i])*1000.0;
            if a.Cells[11,i]<>'' then a1 := a1 + StrToInt(a.Cells[11,i])*100.0;
            if a.Cells[12,i]<>'' then a1 := a1 + StrToInt(a.Cells[12,i])*10.0;
            if a.Cells[13,i]<>'' then a1 := a1 + StrToInt(a.Cells[13,i]);
            if a.Cells[14,i]<>'' then a1 := a1 + StrToInt(a.Cells[14,i])*0.1;
            if a.Cells[15,i]<>'' then a1 := a1 +StrToInt(a.Cells[15,i])*0.01;
        end;
    end;
    Label8.Caption := FloatToStr(a1);
    //统计贷方金额
For i := 1 to a.RowCount-1 do
    begin
        if a.Cells[16,i]='-' then
        begin
            if a.Cells[17,i]<>'' then a2 := a2 - StrToInt(a.Cells[17,i])*10000000000.0;
            if a.Cells[18,i]<>'' then a2 := a2 - StrToInt(a.Cells[18,i])*1000000000.0;
            if a.Cells[19,i]<>'' then a2 := a2 - StrToInt(a.Cells[19,i])*100000000.0;
            if a.Cells[20,i]<>'' then a2 := a2 - StrToInt(a.Cells[20,i])*10000000.0;
            if a.Cells[21,i]<>'' then a2 := a2 - StrToInt(a.Cells[21,i])*1000000.0;
            if a.Cells[22,i]<>'' then a2 := a2 - StrToInt(a.Cells[22,i])*100000.0;
            if a.Cells[23,i]<>'' then a2 := a2 - StrToInt(a.Cells[23,i])*10000.0;
            if a.Cells[24,i]<>'' then a2 := a2 - StrToInt(a.Cells[24,i])*1000.0;
            if a.Cells[25,i]<>'' then a2 := a2 - StrToInt(a.Cells[25,i])*100.0;
            if a.Cells[26,i]<>'' then a2 := a2 - StrToInt(a.Cells[26,i])*10.0;
            if a.Cells[27,i]<>'' then a2 := a2 - StrToInt(a.Cells[27,i]);
            if a.Cells[28,i]<>'' then a2 := a2 - StrToInt(a.Cells[28,i])*0.1;
            if a.Cells[29,i]<>'' then a2 := a2 -StrToInt(a.Cells[29,i])*0.01;
        end
        else
        begin
            if a.Cells[17,i]<>'' then a2 := a2 +StrToInt(a.Cells[17,i])*10000000000.0;
            if a.Cells[18,i]<>'' then a2 :=a2 + StrToInt(a.Cells[18,i])*1000000000.0;
            if a.Cells[19,i]<>'' then a2 := a2 + StrToInt(a.Cells[19,i])*100000000.0;
```

```
            if a.Cells[20,i]<>'' then a2 := a2 + StrToInt(a.Cells[20,i])*10000000.0;
            if a.Cells[21,i]<>'' then a2 := a2 + StrToInt(a.Cells[21,i])*1000000.0;
            if a.Cells[22,i]<>'' then a2 := a2 + StrToInt(a.Cells[22,i])*100000.0;
            if a.Cells[23,i]<>'' then a2 := a2 + StrToInt(a.Cells[23,i])*10000.0;
            if a.Cells[24,i]<>'' then a2 := a2 + StrToInt(a.Cells[24,i])*1000.0;
            if a.Cells[25,i]<>'' then a2 := a2 + StrToInt(a.Cells[25,i])*100.0;
            if a.Cells[26,i]<>'' then a2 := a2 + StrToInt(a.Cells[26,i])*10.0;
            if a.Cells[27,i]<>'' then a2 := a2 + StrToInt(a.Cells[27,i]);
            if a.Cells[28,i]<>'' then a2 := a2 + StrToInt(a.Cells[28,i])*0.1;
            if a.Cells[29,i]<>'' then a2 := a2 +StrToInt(a.Cells[29,i])*0.01;
        end;
    end;
    Label10.Caption := FloatToStr(a2);
end;
```

23.11.2 经验总结

在开发数据库应用程序过程中，如果数据表需要包含时间字段，建议设置该字段的默认值为“GetDate()”，即默认值为服务器的当前时间。当向数据表中插入数据时，在时间字段便可以使用默认值，这样做的好处是在程序中不需要用户选择时间，可以加快用户录入数据的速度，同时减少录入数据出错的机率。

明日科技